



Simply Java Programming

An Application-Driven Tutorial Approach

Java 大学简明教程

—— 实例程序设计



DEITEL

Deitel Deitel

Listfield Yaeger Zhang 著

张琛恩 等译
贺昱曜 审校

Java 大学简明教程——实例程序设计

Deitel Deitel Listfield Yaeger Zhang



本书展示了Java在面向对象、事件驱动等程序设计方面的强大功能。本书出色的教学方式均源于Deitel & Associates公司——该公司拥有众多撰写最畅销编程语言书籍的作者和在100多个国家为超过1 000 000的人讲授程序设计课程的知名教师。

应用程序驱动、手把手的教程式方法——Deitels 将带领读者在一个生动有趣、面向实际的应用程序中探索Java概念。这些实际的应用程序涉及：基于Web的书店 • ATM • 电话号码簿 • 购车还贷 • 付账 • 国旗知识测评 • 筹款募集 • 图形画板 • 清单 • 投资利息 • 工资额计算器 • 微波炉 • 屏幕抓取 • 门禁系统 • 货运中心 • 票务订购 • 打字训练器 • 图形绘制

要点讨论——Java类库 • GUI组件(JButton, JTextField, JList, JTimer, JTextArea, JSpinner, JComboBox, JRadioButton, JMenu, 等等) • 事件处理 • 调试 • 算法 • 伪代码 • UML • 控制语句 • 方法 • 随机数生成 • 数组 • 类 • 对象 • 继承 • 多态 • 接口 • 集合 • 鼠标及键盘事件 • 字符串 • 按序存取文件 • 数据库 • 图形 • 多媒体 • GUI设计 • 三层Web应用程序的开发 • 异常处理 • 迭代器 • ArrayList • Swing • JSP • HTML • GUI程序设计

“本书对于那些喜欢自己动手、实际操作的Java初学者来说简直是太棒了。不必阅读纷繁复杂的语言结构，当你一步一步遵照教程来学习时，便在真实地构建一个个现代的计算机应用程序。”

——Paul McLachlan (Compuware Corporation)

“对于那些想开发面向实际领域编程的学生来说，本书将是最佳选择——书中提供了大量的程序实例及编程过程中解决众多问题的提示。使用Swing开发的应用程序从一开始就贯穿全书，而且更为重要的是，本书一直都在关注问题解决的逻辑。”

——Merrill Parker, Ph. D (Information Systems Technology, Chattanooga State)

“我印象最深的地方就是作者提供了一套近乎完美的练习题，这一定会激发学生学习编程的兴趣。我真心期待在课堂上使用这本书。”

——Ed Weihrauch (Community College of Allegheny County)

“UML技术的无缝集成、面向对象的程序设计、GUI概念的大量使用，这一切对于追求迈向更高级技术的初级程序员来说，是一个最佳的选择。”

——Gavin T. Osborne (Saskatchewan Institute of Applied Science and Technology)

“这是一本可适用于任何Java IDE的好书。有了它，不用再惧怕Java中面向对象程序设计背后那些极具挑战性的概念。它会以一种简单、可读的方式引导学生一步一步地向前走。”

——Catherine Wyman (Senior Professor, DeVry University)

“有位学生缺了几堂关于数组的课程，因而遇到了学习上的困难，我就将正在审阅中的教程8的底稿送给她。后来，她告诉我说已经完全掌握了数组，并对本书内容表现出了极大的热情。”

——Craig W. Slinkman, Ph. D (University of Texas, Arlington)

采用多种教学方法

- 逐步推进的教程式方法为读者展示如何构建并执行完整的应用程序，通过一个模板，引领读者不断完善直至成功
- 采用Deitel独具特色的实时码技术——通过许多完整可执行的应用程序，介绍程序设计的概念
- 采用应用程序驱动的方法——32个教程，145个面向实际的应用程序
- 自测选择题及其答案为读者提供每一小节的及时反馈
- 技术小结，UML活动图
- 调试技术及大量习题教你学会Java的命令行调试程序(JDB)
- 提高程序可靠性、性能及可用性的提示
- 提供大量练习，包括：为每个教程精心准备的10道选择题；两个分别名为“说出这段代码的作用”和“找出代码中的错误”的练习；4道面向实际应用程序的练习，其中包括一个“挑战题”
- 每个教程中的GUI设计导航和Java类库索引，方便读者参考使用
- 每个教程中都包含了一个关键术语列表，本书末尾还附有一个完整的词汇表
- 提供教师用资源（获取方法，详见本书最后所附的“教学支持说明”），包括：PowerPoint幻灯片，所有测试题文档，教师用CD-ROM以及关于本书最新动态的Web网站（www.deitel.com 和 www.prenhall.com/deitel）

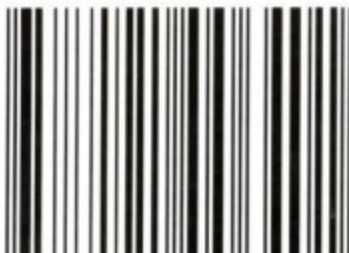


联系作者：deitel@deitel.com

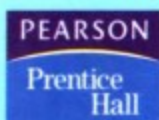
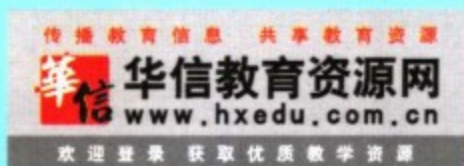
Deitel 全球在线培训：www.deitel.com

免费注册的DEITEL BUZZ ONLINE电子邮件实时通信系统：www.deitel.com/newsletter/subscribe.html

ISBN 7-121-00640-5



9 787121 006401 >



责任编辑：李秦华
封面设计：毛惠庚

本书贴有激光防伪标志，凡没有防伪标志者，属盗版图书

ISBN 7-121-00640-5 定价：72.00元（附光盘1张）

出版说明

21世纪初的5至10年是我国国民经济和社会发展的关键时期,也是信息产业快速发展的关键时期。在我国加入WTO后的今天,培养一支适应国际化竞争的一流IT人才队伍是我国高等教育的重要任务之一。信息科学和技术方面人才的优劣与多寡,是我国面对国际竞争时成败的关键因素。

当前,正值我国高等教育特别是信息科学领域的教育调整、变革的重大时期,为使我国教育体制与国际化接轨,有条件的高等院校正在为某些信息学科和技术课程使用国外优秀教材和优秀原版教材,以使我国在计算机教学上尽快赶上国际先进水平。

电子工业出版社秉承多年来引进国外优秀图书的经验,翻译出版了“国外计算机科学教材系列”丛书,这套教材覆盖学科范围广、领域宽、层次多,既有本科专业课程教材,也有研究生课程教材,以适应不同院系、不同专业、不同层次的师生对教材的需求,广大师生可自由选择 and 自由组合使用。这些教材涉及的学科方向包括网络与通信、操作系统、计算机组织与结构、算法与数据结构、数据库与信息处理、编程语言、图形图像与多媒体、软件工程等。同时,我们也适当引进了一些优秀英文原版教材,本着翻译版本和英文原版并重的原则,对重点图书既提供英文原版又提供相应的翻译版本。

在图书选题上,我们大都选择国外著名出版公司出版的高校教材,如Pearson Education培生教育出版集团、麦格劳-希尔教育出版集团、麻省理工学院出版社、剑桥大学出版社等。撰写教材的许多作者都是蜚声世界的教授、学者,如道格拉斯·科默(Douglas E. Comer)、威廉·斯托林斯(William Stallings)、哈维·戴特尔(Harvey M. Deitel)、尤利斯·布莱克(Uyless Black)等。

为确保教材的选题质量和翻译质量,我们约请了清华大学、北京大学、北京航空航天大学、复旦大学、上海交通大学、南京大学、浙江大学、哈尔滨工业大学、华中科技大学、西安交通大学、国防科学技术大学、解放军理工大学等著名高校的教授和骨干教师参与了本系列教材的选题、翻译和审校工作。他们中既有讲授同类教材的骨干教师、博士,也有积累了几十年教学经验的老教授和博士生导师。

在该系列教材的选题、翻译和编辑加工过程中,为提高教材质量,我们做了大量细致的工作,包括对所选教材进行全面论证;选择编辑时力求达到专业对口;对排版、印制质量进行严格把关。对于英文教材中出现的错误,我们通过与作者联络和网上下载勘误表等方式,逐一进行了修订。

此外,我们还将与国外著名出版公司合作,提供一些教材的教学支持资料,希望能为授课老师提供帮助。今后,我们将继续加强与各高校教师的密切联系,为广大师生引进更多的国外优秀教材和参考书,为我国计算机科学教学体系与国际教学体系的接轨做出努力。

电子工业出版社



教程 1 Moving Shapes 应用程序

介绍计算机, Internet 及 Java 程序设计基础

教学目标

在本教程中, 读者将学到以下内容:

- 标准计算机的组成单元
- 认识低级编程语言和高级编程语言的语言特征
- 高级编程语言的发展历程
- 何谓对象以及对象技术的重要性
- Internet 及万维网的发展
- 第一个 Java 应用程序
- 使用 Internet 获取更多关于 Java 的信息

欢迎来到 Java 世界! 本书采用一种简单易懂、由浅入深的教程式的方法来讲授 Java 编程的基础知识。希望读者在学习如何使用 Java 语言编程的过程中能够更好地领会我们的意图, 同时也在身心上也得到一些愉悦。本书的一个核心是在讲授 Java 的过程中采用了应用程序驱动的方法, 此方法为一些有用的、属于现实世界的计算机应用程序提供了一步一步地进行创建并具备交互功能的具体指导, 而且这种方法同本书特有的实时代码(用于显示大量完整可用的应用程序并利用其输出帮助读者养成良好、扎实的编程技能)技术结合在了一起。此外, 读者还将学习有关图形、多媒体、数据库以及 Web 编程方面的相关技术知识。书中的所有实例均可在随书附带的 CD-ROM 或者是我们的 Web 网站 www.deitel.com 上找到。

目前, 计算机在几乎所有领域里的使用都不断加大。在一个成本增长的时代里, 用来完成计算的成本实际却在大幅度地下降, 这一切都归因于硬件和软件技术的飞速发展。硅片技术的出现, 使计算机变得更加低廉, 以使数以亿计的通用计算机在全世界的范围内正得以使用并在商业、工业、政府以及人们的日常生活中发挥着巨大的作用。

通过阅读本书将会使读者的学习道路充满挑战与回报。如需要与我们交流, 可发 E-mail 至 deitel@deitel.com, 我们将尽快给您做出答复。想了解更多信息, 可以登录我们的 Web 网站: www.deitel.com 和 www.prenhall.com/deitel。

1.1 什么是计算机

计算机是能够以超过人类百万甚至是十亿倍的速度执行计算并做出逻辑判断的一种设备。例如, 当今许多功能最为强大的个人计算机能够以每秒几十亿次的速度计算。某位操作台式计算机的人可能需要花费一生的时间才能够计算出相当于一个强大功能的个人计算机在一秒之内所完成的计

算数量。今天，最快的超级计算机每秒能够执行千亿次的计算。每秒计算万亿次指令的计算机已经在所研究的实验室中投入工作了。

计算机是通过使用称之为计算机程序的指令集来对数据进行处理。这些程序可以指导计算机有序地通过由计算机程序员所指定的操作集。在Java及其他编程语言中所使用的面向对象的编程技术（Object-Oriented Programming, OOP）（能够将现实世界中的对象用软件中的对等物进行模拟）是一个非常关键的突破，它能够极大地提高程序员的创造力。本书中，通常使用“应用程序”而不采用“程序”这个术语。应用程序是一个用于完成某种特定任务的程序。绝大多数的教程中将会给出5个有关面向对象的应用程序——1个出现在主要实例中，其余4个则在习题中出现。这样，加起来本教程中将总共包括145个应用程序。

计算机是由称之为硬件的不同设备（如键盘、屏幕、鼠标、硬盘驱动器、存储器、CD-ROM驱动器、处理单元）所组成的。而在计算机中运行的程序通常指的是软件。

自测题

1. 计算机通过使用称之为 _____ 的指令集来对数据进行处理。
a) 硬件 b) 计算机程序 c) 处理单元 d) 程序员
2. 用来组成计算机的设备被称之为 _____。
a) 硬件 b) 软件 c) 程序 d) 应用程序

答案：1) b 2) a

1.2 计算机的组织结构

计算机可以分解成6个不同的单元：

1. **输入单元（Input unit）** 这属于计算机的“接收”部分，它用于从各种不同的（如键盘和鼠标）输入设备中获取信息（数据和计算机程序）。输入设备还包括麦克风（将语音记录到计算机中）、扫描仪（用于扫描图像）以及数字照相机（用于拍摄照片、创作图像视频）。
2. **输出单元（Output unit）** 这属于计算机的“运输”部分，它能够取出计算机所处理的信息并将其放在各种不同的输出设备上，这些信息可在计算机的外部使用。其输出可在屏幕上显示、在音频/视频设备上播放、在纸张上打印并通过计算机网络（如Internet）发送出去，等等。输出还能控制其他的设备，如制造业中所使用的机器人。
3. **存储单元（Memory unit）** 计算机的这个可用于快速访问但相对来说低容量的“仓库”部分，会临时存储某个应用程序运行时的数据。存储单元保留来自输入设备的信息，因而这些信息会及时地得到处理。为了能够执行，计算机程序必须处于存储器当中。存储单元将一直保留所处理的信息直到该信息被发送到输出设备上，在那里用户便可以使用了。通常，存储单元被称为内存或是主存。随机访问存储器（Random access memory, RAM）就属于主存。主存通常是易失的，这意味着当计算机关闭时，数据会被擦除。
4. **算术及逻辑单元（Arithmetic and logic unit, ALU）** ALU属于计算机的“制造”部分。它可完成如加、减、乘、除等运算，也能够做出允许计算机完成诸如判断内存中两个项目是否相等的断定。
5. **中央处理单元（CPU）** CPU作为计算机的“管理”部分，会监控其他部分的操作。一旦有信息需要被读入存储单元，CPU便会通知输入单元做好准备，并指挥ALU何时应在计算中使用来自存储单元的信息并通知输出单元何时将存储单元中的信息发送至不同的输出设备上。

6. 二级存储单元 (Secondary storage unit) 这部分是计算机的一个长效、高容量的“仓库”。二级存储设备,如硬盘驱动器、DVD驱动器、CD-ROM驱动器、Zip驱动器以及软盘驱动器,通常存储其他单元当前所不使用的程序和数据;计算机在需要使用的时候能够检索出这个信息——这个时间有可能是几个小时、几天、几个月、甚至是若干年以后。二级存储设备中的信息在访问上要比主存中存储的信息花费更长的时间。然而,二级存储设备在价钱上比主存便宜很多。二级存储器通常也是非易失的,即当计算机关闭以后仍就能够保留信息。

自测题

1. _____ 用于执行计算并具备决策的机制。
a) 中央处理单元 b) 存储单元 c) 算术及逻辑单元 d) 输出单元
2. 当计算机关闭时,在 _____ 中存储的信息通常会被擦除。
a) 主存 b) 二级存储器 c) CD-ROM 驱动器 d) 硬盘驱动器

答案: 1) c 2) a

1.3 机器语言、汇编语言和高级语言

程序员在使用各种不同的编程语言编写计算机所执行的程序指令时,其中有些语言可被计算机直接理解,而另一些则不能够被计算机直接理解,需要一些中间的转换步骤。尽管今天仍有上百种计算机语言正在使用,然而按照所呈现的不同形式可以将它们划分成三种通用类型:

1. 机器语言
2. 汇编语言
3. 高级语言

计算机能够直接理解其自身的机器语言。作为某种特定计算机的“自然语言”,机器语言由计算机的硬件设计所决定。机器语言(也称为第一代语言)通常是由指挥计算机处理最基本的操作的数字流(最终为多个的0和1)组成的。机器语言是与机器相关的语言,这就意味着特定的机器语言只能运行在某种特定类型的CPU之上:

```
+1300042774
+1400593419
+1200274027
```

利用机器语言编程的缺点是速度慢、易出错。除了使用计算机能够直接理解的数字字符串以外,程序员还可以通过类似英文的一些缩写来表示该计算机的基本操作。这些缩写便构成了汇编语言(同时也称为第二代编程语言)的基础。被称之为汇编程序的翻译器程序会根据计算机的速度将汇编语言程序转换成机器语言。下面的这段汇编语言程序,同样是把加班费同基本工资值相加,然后将结果存储到薪金总额之中,然而对于阅读人员来说,所给出的步骤多少要比使用机器语言的那个例子更为清晰:

```
LOAD  BASEPAY
ADD   OVERTIMEPAY
STORE GROSSPAY
```

这段汇编语言代码对人们来说是更为清晰的,但只有通过汇编程序将其翻译成机器语言时,计算机才可能理解它。

尽管汇编语言比起机器语言能够让程序员更快地编写程序,但即便是一些最为简单的任务,汇编语言仍需要较多的指令才可以完成。为了加速程序开发的过程,程序员主要使用的是某些高级语言(同时也被称为第三代语言),这时,一条程序语句就可以完成相当多的任务。被称为编译程序的翻译器程序可将高级语言程序转换成机器语言。高级语言使得程序员所编写出来的指令看起来像是些日常的英文,并且还可包含一些常用的算术标记。例如,利用高级语言编写的工资单应用程序可能会有一条这样的语句:

```
grossPay = basePay + overTimePay
```

从这个例子中就很明显地能够看出来程序员为什么喜欢使用高级语言而不是机器语言或者是汇编语言。Java便是世界上最为流行的高级编程语言之一。下一节,读者将了解到有关Java的起源及使用它所带来的益处。

由于把一个高级语言程序编译成机器语言的过程可能会花费相当多的机器时间,因此便开发出了解释器程序,它可直接地执行高级语言程序而无需将这些程序编译成机器语言。尽管被编译的程序比解释后的程序在执行速度上更快,但在某些开发环境中使用解释器程序仍然是很普遍的,因为开发环境中的程序由于频繁地添加新的特征并改正一些错误,因此就需要重新进行编译。一旦程序开发出来以后,只需产生一个得到编译后的版本便能以更为高效的方式运行了。

今天,第四代语言(fourth generation languages, 4GL)的出现在使用上更加接近如英文这样的自然语言。第四代语言主要在称为数据库的数据组织集合中用来管理所存储的信息。比如,使用第四代语言开发的一个自动取款机(ATM)应用程序可能会包含一条这样的语句:

```
SELECT balance  
  FROM AccountInformation  
 WHERE accountNumber = "123456"
```

作为取得指定账号余额来使用,或者是通过语句:

```
UPDATE AccountInformation  
  SET balance = 365.74  
 WHERE accountNumber = "123456"
```

更新指定账号内的余额。读者将在教程 26 中了解到更多有关数据库和第四代语言(称为 SQL)的知识。

自测题

1. 计算机 CPU 惟一能够直接理解属于自己的编程语言是 ____。
a) 高级语言 b) 汇编语言 c) 机器语言 d) 英文
2. 可将高级语言程序翻译成机器语言的程序被称为 ____。
a) 汇编程序 b) 编译程序 c) 程序员 d) 转换器程序

答案: 1) c 2) b

1.4 Java 概述

Java 语言是在 1991 年由 James Gosling 创立的,当时是作为 Sun Microsystems 公司一项合作研究计划中的一部分。这个代码名为 Green 的项目开始是为预测微处理器(能够使计算机进行工作的芯片)是否将对智能型的消费类电子设备产生巨大的影响。Java 编程语言是以当时正广为使用的,能为面向对象编程提供不同功能的 C++ 语言作为基础的。

智能型消费类电子设备的市场,其发展过程并没有如Sun公司所预料的那样迅速。结果,这个Green项目建设初期便遇到了一些的困难。幸运的是,万维网在1993年突然间流行了起来,Sun公司看到使用Java创建动态内容(具有动画及交互功能的内容)的Web页面在当时是很有潜力的。为此Sun公司便引入了一个称之为applet的Java程序的创建机制,用以在Web网页上执行并通过使用Web浏览器进行显示。

1995年5月,Sun公司在其技术大会上正式宣布了Java。通常,类似这样的事件可能不会引起太多的注意。可是,由于人们对Web兴趣的空前高涨,使得Java在企业界中立即引起了震动。时至今日,开发人员使用Java创建动态的Web网页、构建大型的企业级应用程序、提升服务器(一旦用户浏览Web站点便会将相关内容发布到各自浏览器所在的计算机上)的性能、为一些消费类设备(如手机、呼机、PDA)提供应用程序和其他的一些用途。Java不再只是将Web网页“活跃起来”的一门语言了,它已成为满足许多机构编程时的首选语言。

如此多的程序员愿意使用Java语言这的确是有一些原因的。首先,Java是完全面向对象的。对象是可复用软件中的组件,它能模拟现实世界中的对象。面向对象的程序通常要比利用以往技术所开发的程序更容易理解、更正和修改。

其次,Java程序是由一些可用来定义对象,称为类的部件组成的。类中包括一些用于完成任务的方法,并可在它们完成这些任务时返回有用的信息。我们可以创建所需要的每一个部件来构建Java程序。然而,大多数Java程序员会利用Java类库中现存类的丰富接口,这就是人们常说的Java API(Application Programming Interface,应用程序编程接口)进行程序的构建。因而,关于Java“世界”的学习实际上存在着两个部分的内容。一方面是Java语言本身,目的是编写出属于自己的类和方法;另一方面,便是要学习Java类库,它提供了可复用的类并且作为可扩展的集合。本书中,我们将对类库中的许多类进行讨论。读者将使用其中的一部分来创建含有图形用户界面(graphical user interfaces, GUI)的应用程序,它们是用户同应用程序进行交互的可视化部分。读者还将使用类库创建自己GUI的事件驱动,以便能够响应诸如鼠标点击、键盘按键等由用户所启动的事件。

最后,许多程序员钟爱Java的一个原因是因为它的平台无关性,这是指,利用Java所开发的应用程序能够在各种不同的计算机平台(即运行在各种不同操作系统如Windows, Linux之上的不同类型的计算机)上创建和运行。“编写一次,到处运行”的概念真正让程序设计人员感受到了可移植性——应用程序可被位于不同计算机平台上的人们所使用,而无需考虑程序创建时的计算机平台。由于Java巧妙地将其程序的编译和解释执行结合到了一起,因而能够获得这样的可移植性及良好的执行性能。

Java已成为实现基于Internet的应用程序及通过网络进行通信的设备的首选语言。若有一天我们家中的某个新型立体声设备或其他什么设备被Java技术连接到网络上,完全不必大惊小怪!即便是今天,像手机、寻呼机、个人数字助理(Personal Digital Assistants, PDA)这样的无线设备正在使用着某种基于Java的网络应用程序,并通过所谓的无线Internet进行通信,读者会在本书中学到这些知识。

自测题

1. Java语言是在1991年由James Gosling在_____创立的。
a) Apple b) IBM c) Microsoft d) Sun Microsystems
2. 使用_____语言所创建的应用程序能够在多种运行不同操作系统的计算机平台上运行。
a) GUI驱动的 b) 事件驱动的 c) 独立于平台的 d) 与平台相关的

答案: 1) d 2) c

1.5 其他高级语言

尽管出现过上百种高级语言,但只有少数的一部分语言得到了广泛应用。IBM公司在20世纪50年代中期开发的FORTRAN (FORmula TRANslator, 公式转换语言)语言为的是能够创建有关科学及工程方面的应用程序,因为这些应用程序需要有复杂的数学计算。FORTRAN仍旧在工程界被广泛应用。

COBOL (COmmon Business Oriented Language, 面向商业的通用语言)是在20世纪50年代后期由一批同政府及商用计算机用户合作的计算机制造商开发的。COBOL主要是用在管理具备大量数据的商用应用程序中的。如今商用软件中相当大的一部分仍然是由COBOL编写的。

20世纪70年代早期,Dennis Ritchie在贝尔实验室里开发出了C语言,因其被作为UNIX操作系统的开发语言而获得了广泛认可。C++——一个C语言的扩充版,同样是在贝尔实验室里由Bjarne Stroustrup于20世纪80年代早期开发出的。C++为面向对象的程序设计提供了许多功能。今天,许多主要的操作系统都是用C或者C++编写的。

BASIC (Beginner's All-Purpose Symbolic Instruction Code, 初学者通用符号指令码)编程语言是在20世纪60年代中期由Dartmouth学院的两位教授Joho Kemeny和Thomas Kurtz,针对编写简单的程序而开发的一种语言。BASIC的主要目的是让初学者熟悉有关编程方面的技术。1991年微软引入了Visual Basic,利用它可简化微软窗口应用程序的开发过程。

Visual Basic .NET是专为微软最新的.NET框架而设计的。Visual Basic .NET提供了完全的面向对象的方法,并使用称之为框架类库(Framework Class Library, FCL)的这个可复用软件组件的强大类库。该类库可以在Visual Basic .NET, Visual C++ .NET, C#及微软同其他一些软件厂商合作提供的针对.NET支持的语言之间进行共享。

微软的C#编程语言是专为.NET框架而设定的。它源于C, C++和Java。C#, Java和Visual Basic .NET具有一些相似的、可比较的功能,这可能会为读者学习Java创造很多的机会。

自测题

- _____是C的一个扩展且提供了面向对象的能力。
a) Visual Basic b) C++ c) assembly language d) Windows
- 程序设计语言_____最初是为微软的.NET框架而开发的。
a) C# b) Java c) C++ d) Visual Basic
- 20世纪50年代后期开发的_____,今天仍用于创建商业软件中的相当大的一部分。
a) COBOL b) FORTRAN c) Java d) C
- 20世纪50年代开发的_____现在仍在用来创建需要有复杂数学计算的科学及工程的应用程序。
a) Visual Basic b) FORTRAN c) COBOL d) C#

答案: 1) b 2) a 3) a 4) b

1.6 结构化程序设计

整个20世纪60年代,在开发软件的过程中所做的工作常常落后于预先的计划,而成本却大幅度地超过了预算,其结果则是最终的产品并不是可靠的。人们开始认识到软件开发活动的复杂性远远超过了他们所设想的复杂程度。试图阐明这些问题而做的一些研究性工作导致了一场结构化编程的革命,结构化编程可作为创建清晰、正确以及易于修改软件的一种具有一定规则的方法。

这个研究的一项结果便是在1971年开发的Pascal编程语言。该语言是以17世纪数学家及哲学家Blaise Pascal的名字命名的,为的是讲授结构化的编程思想,而它很快就成为了多数大学里面所选用的一门用于初级编程课程的语言。遗憾的是,该语言缺乏可应用于工商业及政府部门应用软件的许多特性。相比之下,同样也是来自有关结构化编程研究的C语言却没有Pascal语言的这些限制,许多专业级的程序员因而很快便采纳了它。

20世纪70年代及20世纪80年代早期出现的Ada编程语言是在美国国防部的主持下创立的。该语言是以诗人拜伦的女儿Ada Lovelace的名字命名的。Ada Lovelace通常被公认为是世界上的第一位计算机程序员,因为她曾在19世纪早期为Charles Babbage设计的机械式计算设备中的分析机编写了一个程序。

自测题

1. _____ 是用来在学术环境中讲授结构化编程思想的。
a) C++ b) C c) Java d) Pascal
2. _____ 通常被公认为世界上的第一位计算机程序员。
a) 拜伦 b) Dennis Ritchie c) Ada Lovelace d) Charles Babbage

答案: 1) d 2) c

1.7 诠释软件的发展方向: 对象技术

20世纪70年代以来,伴随着人们逐渐对结构化编程思想所带来好处的认识,开始出现经过改进后的软件技术。然而,直到面向对象的编程思想在20世纪80年代和20世纪90年代被广泛地使用以后,软件开发人员才最终意识到他们能够使用一些工具来极大地改进软件开发的过程。

究竟什么是对象、它们到底有什么特别的地方?对象技术实际是一个用于创建有意义的软件单位的包装模式。如日期对象、时间对象、视频对象、文件对象、记录对象,等等。事实上,几乎任何名词都可以合理地表达成软件的对象。对象还拥有一些属性(也被称为特性),如颜色、大小和重量;并且能够执行一些操作(也称为行为或方法),如移动、睡眠或绘制图形,等等。类表示的是相似对象的一种类型。例如,所有的小汽车均属于car类,尽管不同的小汽车可能在制造、型号、颜色和所选用的工具包上有所不同。类确定了其对象的通用形式以及可作用于该类对象上的属性和操作。对象和其类的关系在很大程度上可近似为某个建筑物和它的蓝图之间的关系。

在面向对象的语言出现以前,过程化的编程语言(如FORTRAN, Pascal, BASIC, C)强调的是操作(动词)而非对象(名词)。使用当今流行的面向对象语言,如Java, Visual Basic .NET, C++和C#,程序员可以采用一种更为自然的、能反映出他们所观察世界的面向对象的方式进行编程。

利用对象技术,正确设计的类可在以后的工程中进行复用。使用类库能极大地降低因实现新的应用程序而带来的大量冗余工作。某些组织报道说,软件的这种复用性实际并不是面向对象编程带来的最为显著的优点。他们认为,通过面向对象编程技术产生的软件更容易被理解,因为软件被较好地组织在一起,且无需过多地维护。

面向对象使程序员的注意力集中到了“宏图”上。而不必去担心可复用对象是如何实现的这些微小的细节问题,程序员还可以集中于对象之间的行为和它们的交互式上。一幅显示所有树木、房屋和车道的地图很可能是难以阅读的。当这些细节被删除而只保留重要的信息(道路)时,则地图会变得更加容易理解了。同样,应用程序可以被划分为易于理解、修改,以及可更新且已隐藏了这些细节的对象。很明显,面向对象的编程技术将成为至少下一个10年之内的主要的程序设计方法。Java语言就是现在世界上最广为使用的面向对象的编程语言之一。

自测题

1. _____ 强调的是操作（动词）而不是事物（名词）。
a) C# b) 面向对象的编程思想 c) Java d) 过程化的编程思想
2. 在面向对象的编程技术中，作为相似对象的一种类型，_____ 在某种意义上来说类似于蓝图。
a) 类 b) 属性 c) 行为 d) 特性

答案：1) d 2) a

1.8 Internet 与万维网

20 世纪 60 年代末，ARPA [Advanced Research Projects Agency of the Department of Defense, (美国)国防部高级研究计划署]设想将大约十多所由 ARPA 资助的大学和研究机构中的主机系统连接成一个网络。这些计算机准备连接到当时看来已是了不得了的以 56 Kbps (1 Kbps = 1024 位/秒) 速度进行工作的通信线路上。那时，多数人（甚至很少有人连接过网络）一直是通过每秒 110 比特速度的电话线来连接计算机的。学术方面的研究似乎总是能够走得更远。ARPA 不断地进行完善，很快就被人们称为 ARPAnet，可以认为它就是当今 Internet 的祖父。

事情总是与最初的计划有所不同。尽管 ARPAnet 可以将研究人员的计算机连成网络，但事实证明它的主要优点只是通过现在为人们所熟知的电子邮件 (E-mail) 这样的工具来完成快速和简便的通信功能。即便今天的 Internet 也仍然是这样的，利用 E-mail、即时信息和文件传输可让全世界数以亿计的人们彼此之间进行通信。

通过 ARPAnet 进行通信的协议（换句话说，即一些规则的集合）就是后来的传输控制协议 (Transmission Control Protocol, TCP)。TCP 保证把称之为“包”的部件组成的信息正确地从一个发送者那里路由到接收者一方，并保证这些信息在到达时是完整的。

在 Internet 的早期变革中，世界上的许多组织同时也在为组织内（在一个组织的内部）及组织间（在多个组织之间）的通信实现着其自身的网络。大量不同的有关网络连接的硬件和软件便出现了。其中的一个难题便是如何让这些不同的网络彼此之间进行通信。ARPA 通过发展 Internet 协议 (Internet Protocol, IP) 实现了这个目的，利用这样的协议创建了一个真实的“网络中的网络”，它就是当前 Internet 的体系结构。这个组合在一起的协议集通常被称为 TCP/IP。

许多企业很快便认识到通过使用 Internet 可以改进工作并为客户提供更新和更优质的服务。一些公司开始花费大量资金开发并改善现有的 Internet。这给通信公司以及硬件和软件提供商们为满足日益增长的基础设施的需求创造了一个强烈的竞争环境。结果导致 Internet 上带宽（通信线路上信息的承载量）的显著增加，而硬件成本却在大幅度下降。

万维网 (World Wide Web, WWW) 是同 Internet 相关的硬件和软件的一个集合，它能够让计算机用户定位并浏览几乎任何主题的基于多媒体的文档（将各种文本、图像、动画、音频、视频组织在一起的文档）。尽管 Internet 是在 30 多年以前开创的，但万维网的出现相对来说确实是最近的事情。1989 年，任职于 CERN（欧洲原子核研究组织）的 Tim Berners-Lee 开始研发一项通过使用“超链接”文档进行信息共享的技术。Berners-Lee 将他的发明称为超文本标记语言 (HyperText Markup Language, HTML)。他同时也编写了用来构成这个新的超文本信息系统骨架的一些协议，并将其引述为万维网。

1994 年 10 月，Berners-Lee 创立了一个名为万维网联盟 (World Wide Web Consortium, W3C, www.w3.org) 的组织，致力于万维网技术的开发。W3C 的一个主要目标便是让全世界的 Web 都可以进行访问——不论个人是否拥有资质，使用什么样的语言，或者不同的文化差异。

Internet 和万维网一定能列入人类最重要的发明创造物之列。过去,大多数计算机应用程序是在“独立”的计算机(未连接其他计算机的计算机)中运行的。而今天的一些应用程序则专门是为世界范围内数以亿计的计算机间的通信来编写的。Internet 和万维网甚至能够让某些个人及小规模的公司暴露给全世界。它们震撼般地改变着我们所从事的商业活动以及支配着我们个人的生活方式。为了突出 Internet 和 Web 编程的重要性,本书将在后面包括四个教程,通过这些教程读者会实际地创建并运行一个基于 Web 的网上书店应用程序。有趣的是,Java 的平台无关性实际上是作为早期通过 Java 进行 Web 开发的结果才得以出现的。

自测题

1. 今天的 Internet 是从当时(美国)国防部的一个称为 _____ 的项目发展而来的。
a) ARPAnet b) HTML c) CERN d) WWW
2. 在 Internet 上用来通信的组合协议集常被称为 _____。
a) HTML b) TCP/IP c) ARPA d) TCP

答案: 1) a 2) b

1.9 Java 运行环境

本书曾在“准备工作”(位于本教程之前)中介绍了有关 Java 2 软件开发包(J2SDK)的安装。本节中,读者将学习 Java 运行环境(Java Runtime Environment, JRE),它是 J2SDK 中能够用来执行(运行)Java 应用程序的一部分。而整本书中的大部分内容,读者都将集中在两种类型: .java 和 .class 文件的学习上。使用扩展名为 .java 的文件(称为源代码文件)来存储作为一名程序员身份的读者所编写的一些 Java 语句。这些语句可用于表示应用程序执行的操作。读者已在 1.3 节中了解到,这样的一些语句是不会被计算机理解的。为执行这个应用程序,在 .java 文件中存储的语句必须被转换成 JRE 所能理解的一些语句。将这些语句从一种高级语言转换成某种机器语言的过程便是编译。在教程 2 中,读者会了解到如何将一个 .java 文件编译成 .class 文件。尽管程序员是难以理解包含在 .class 文件中的这些语句,但它们却能够被 JRE 执行。下一节,我们将会执行一个 .class 文件。

命令提示符是通过提示符(见图 1.1 中的 C:\>)键入文本以输入计算机指令的 Windows 程序。然而,在一个如 Windows 的图形用户界面中,可以双击记事本图标来启动记事本文本编辑器,还可利用一个命令提示符键入 Notepad 并按下 Enter 键来载入这一程序。

图 1.1 中显示了一个运行 Windows 2000 的命令提示符窗口。Windows 2000 的命令提示符窗口位于 Start → Programs → Accessories 的菜单下。当命令提示符窗口启动以后,初始目录为 C:\。



图 1.1 Windows 2000 中的一个命令提示符窗口

图 1.2 是一个运行 Windows XP 的命令提示符窗口。可通过 Start → All Programs → Accessories 菜单来访问 Windows XP 的命令提示符窗口。当命令提示符窗口启动以后,初始目录为 C:\Documents and Settings\Administrator。Administrator 是该台计算机的用户名。读者若使用的是自己的计算机,则 Administrator 将表示为此机器的用户名。

Windows XP
的初始目录



图 1.2 Windows XP 中的一个命令提示符窗口

如果是 Windows 98 或者是 Windows Me, 命令提示符则被称为 MS-DOS 提示符, 那么起始的目录也会是不同的。

自测题

1. 一个 _____ 文件可包含由读者编写的 Java 语句。

- a) .java b) .jre c) .exe d) .class

2. Java 运行环境只可以识别出 _____ 文件。

- a) .java b) .jre c) .exe d) .class

答案: 1) a 2) d

1.10 新手上路: Moving Shapes 应用程序

本节将介绍一个 Java 应用程序, 采用特有的应用程序驱动的方法。以后在每个教程中, 读者都将以某个应用程序的“探试”作为开始, 而实际上, 读者是用一个完整的应用程序来运行和交互的。当进行完“探试”之后, 读者会学到一些用于构建此应用程序的 Java 特性, 其最终目的是将所学到的知识“融合在一起”以创建属于自己的应用程序。在教程 1 中, 读者将运行一个“绘画”应用程序, 通过运行它可以绘制出不同的可在屏幕上进行移动的图形。读者还将在教程 27 中实际地创建一个类似的应用程序。

我们创建这个 Moving Shapes 应用程序为的是演示一个完整的图形及交互式的应用程序。在下面探试 Java Moving Shapes 应用程序中, 读者将运行该程序并自行加入运动的图形。本应用程序中的这些元素和功能将是读者在本课程中学习编程时的一些典型内容。



探试 Java Moving Shapes 应用程序

1. **检查路径** 请阅读“准备工作”之中的内容, 确保计算机中已正确设置了 Java, 并且也已将书中的实例复制到了硬盘驱动器中。
2. **定位到完整的应用程序中** 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\Examples\Tutorial01\MovingShapes` 然后按下 Enter 键 (如图 1.3 所示), 将目录改变到这个完整的 Moving Shapes 应用程序的目录之下。命令 `cd` 可用来改变目录。

注意当前
目录已改变



图 1.3 将目录更改为 Moving Shapes 应用程序的目录

3. **运行 Moving Shapes 应用程序** 现在, 正处在包含应用程序的目录下, 键入命令 `java MovingShapes` (如图 1.4 所示) 并按下 Enter 键来运行这个应用程序。利用命令 `java`, 后跟该应用程序的 .class 文件的名字 (本例中, 为 `MovingShapes`), 便能够执行该程序。注意在使用 `java` 命令的时候无需指定 .class 这个扩展名。随后此应用程序的执行结果将显示在屏幕上 (如图 1.5 所示) [注意: Java 中的命令都

是大小写相关的。在键入应用程序的名称时，包含一个大写的 M（位于 Moving 中）和一个大写的 S（位于 Shapes 中）是很重要的。否则，系统将不执行该应用程序】。

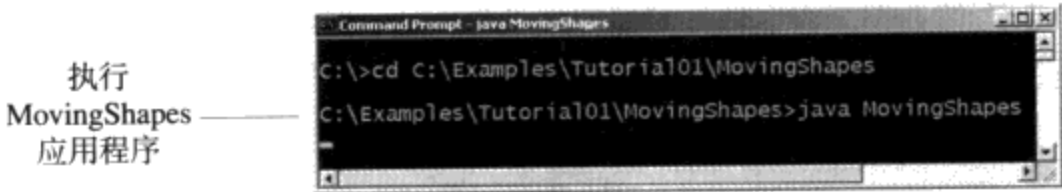


图 1.4 运行 Moving Shapes 应用程序

在图 1.5 中，对一些被称为 GUI 组件的图形元素均进行了标注。这些 GUI 组件包括一个 JButton，一个 JComboBox，一个 JPanel 以及一个 JFrame。JFrame 是一种窗口，它可包含其他的组件并用作该应用程序的显示窗口。读者使用应用程序中的这些 GUI 组件可以指定所绘制图形的类型（使用 JComboBox 进行选择）及颜色（使用 JButton 进行选择）。

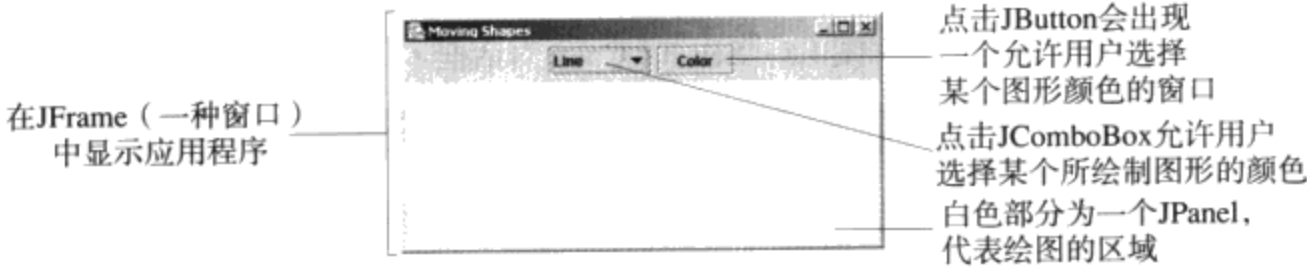


图 1.5 具备一个可交互 GUI 的 Moving Shapes Java 应用程序

通过使用现有的这些 GUI 组件（它们都是一些对象），便可利用 Java 得到可运行的且功能强大的应用程序，这要比自己编写所有的代码快得多。本书中，读者将学到如何使用许多现有的 GUI 组件，以及如何去编写用于自定义属于自己的应用程序的代码。

当应用程序开始执行时，在 JComboBox 中显示的图形为 Line，且所绘制的图形颜色为灰色（通过 Color 按钮的背景颜色来表示）。这些值被认为是默认的——即首次运行程序时所看到的初始值。这些默认值表明，如果用户在不选择某个图形或某种颜色便开始绘图的话，则使用灰色的线条来进行绘制。通过选择所绘制图形的类型及颜色，便可以同该应用程序进行交互了。

4. 改变绘制图形的类型 点击 Line 这个 JComboBox，会显示出一个选项列表（如图 1.6 所示）。当鼠标移动至不同的选项上时，注意到鼠标经过的每个选项会被突出显示。点击 Rectangle 选项。在 JComboBox 中显示的文字将改变为 Rectangle（如图 1.7 所示）。

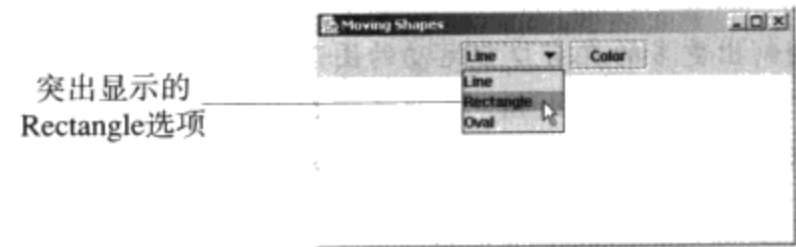


图 1.6 显示不同选项的 JComboBox

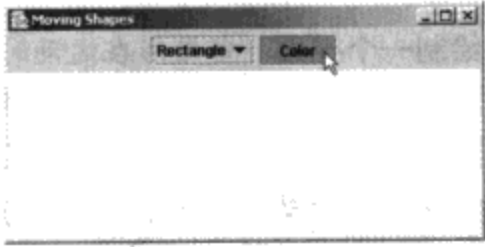


图 1.7 点击 Color 这个 JButton

5. 显示 Select a Color 对话框 点击 Color 这个 JButton（如图 1.7 所示）。出现一个对话框（如图 1.8 所示）（对话框是用来向用户显示一条信息或者是显示可用于选择各种不同选项的一个窗口）。此处这个特定的对话框是由 Java 预先定义好的，通过它可允许用户在自己的应用程序中选择多种颜色。
6. 改变绘制图形的颜色 通过点击颜色方块（也称为样片）里的一个颜色从调色板（连续的颜色）中选取某种颜色。一旦选择了某种颜色以后，可按下 OK 按钮。此时对话框将关闭，而按钮的背景颜色则改变为所选择的颜色（如图 1.9 所示）。
7. 绘制矩形 通过在绘图区域的上方拖拽鼠标进行图形的绘制。一旦有图形被绘制出来，该图形将不停地在 JPanel 中进行运动。把鼠标放置在绘图区域中，然后拖拽鼠标便可绘制出一个矩形框图（如

图 1.10 所示)。拖拽鼠标的做法是，按住鼠标左键进行移动。然后，再松开鼠标按键。这个矩形便开始在绘图区域的内部来回运动了（如图 1.11 所示）。而图形的方向和速度是由应用程序随机选择的。矩形将不停地运动直到第 10 步关闭该应用程序时为止。

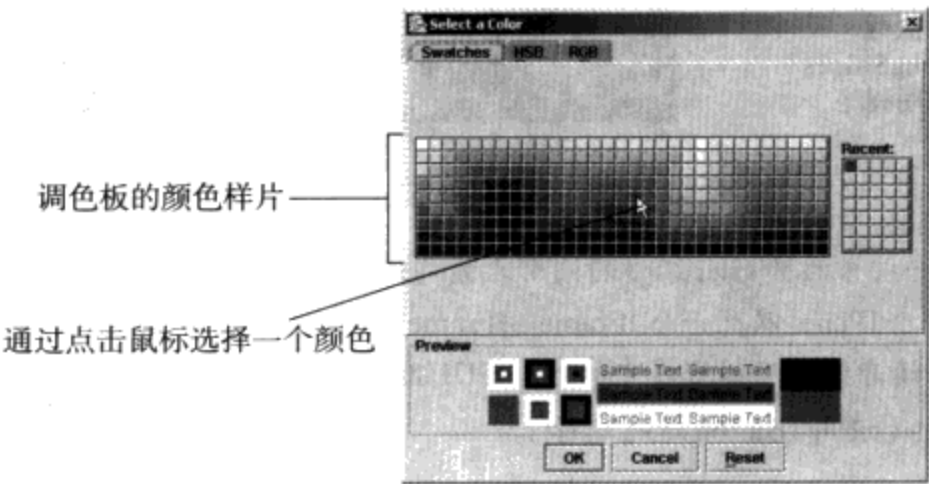


图 1.8 Select a Color 对话框

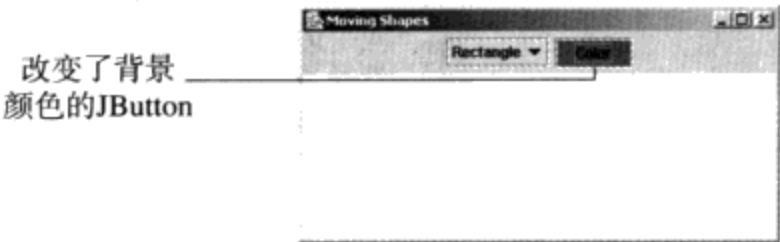


图 1.9 颜色已得到修改的应用程序

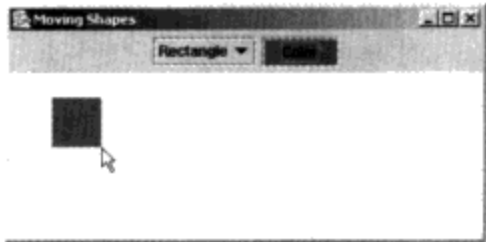


图 1.10 绘制一个矩形

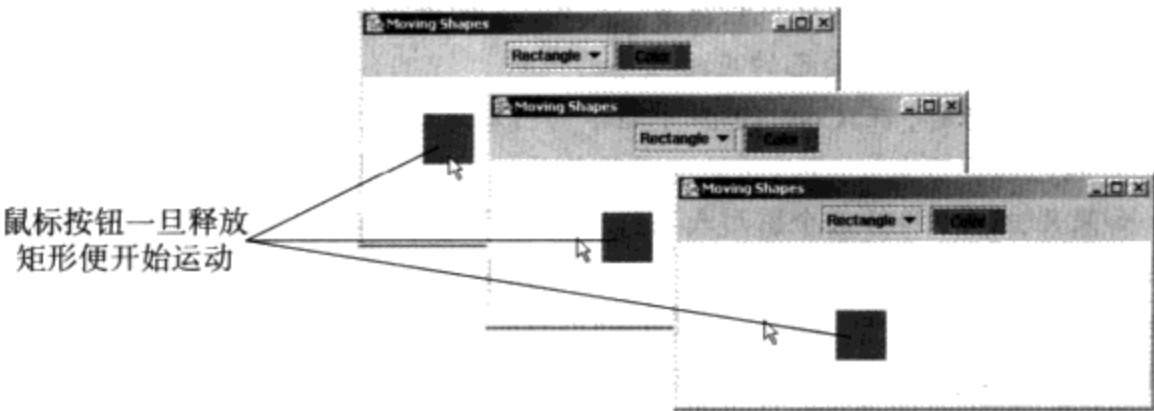


图 1.11 鼠标按钮释放以后矩形便开始运动

8. 绘制一个椭圆 当矩形在运动时，还可绘制出更多的在窗口中运动的图形。点取上面的 JComboBox，选择 Oval（如图 1.12 所示）。然后再点击 Color 这个 JButton，为自己的这个图形选择一种新的颜色。现在，在绘图区域的上方拖拽鼠标便可以绘制出一个椭圆。鼠标按钮释放以后，这个椭圆同样会在窗口中进行运动（如图 1.13 所示）。

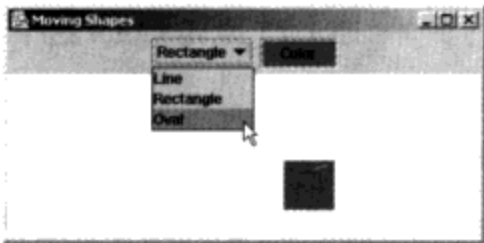


图 1.12 选择 Oval 作为要绘制的图形

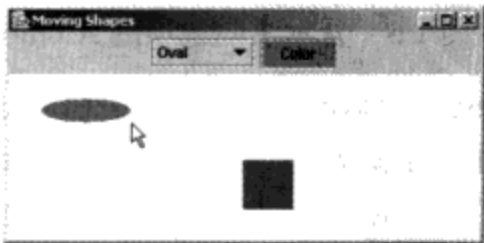


图 1.13 绘制一个椭圆

9. 绘制其他图形 创建其他更多的一些图形。读者可以继续绘制一些矩形和椭圆，同时也可以从 JComboBox 中选择 Line 来绘制一些线条。绘制出的这些图形都将在 JPanel 中进行运动。注意，当两个图形彼此运动到相同的位置上时，其中一个图形将覆盖另外的一个图形，最后绘制的图形会位于最上方。

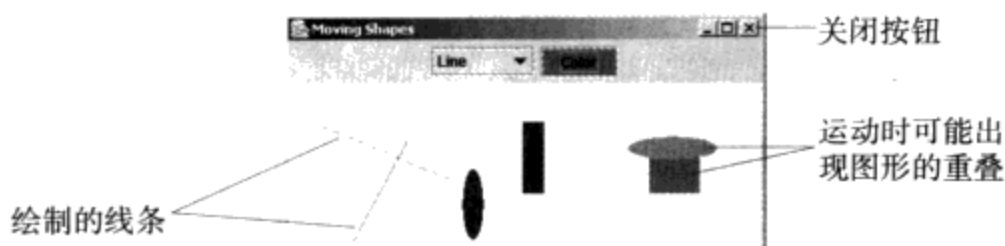


图 1.14 绘制多种图形的 Moving Shapes 应用程序

10. 关闭应用程序 点击关闭按钮 关闭应用程序。返回到命令提示符窗口下。

11. 关闭命令提示符 点击命令提示符窗口的关闭按钮将其关闭。

这些允许同应用程序进行交互的组件 (JButton, JComboBox, JPanel, JFrame) 已经作为 Java API 的一部分被定义好了。读者可按照自己的用途自定义这些组件。例如, 本应用程序中的 JComboBox 被自定义为可提供三条选项——Line, Rectangle, Oval。JButton 被自定义为可显示文字的颜色。读者不需要定义这些组件的通用外观。这就是软件复用——一个优秀的例子。Sun 公司已经提供了这些组件——别的公司就不必花费大量的时间和金钱来创建它们了, 这样, 世界范围内的数以百万的程序员就可以同读者一样, 使用它们来创建具备专业化质量的 GUI, 而根本不需再做更多的编程了。

1.11 Internet 及 Web 资源

在 Internet 及 Web 上能够找到大量有用的资源。本节中将包括一些有趣的和拥有丰富资料的 Web 站点。这些站点的所有链接均包含在随书附带的 CD-ROM 以及 www.deitel.com 网站上。类似本节的这些可参考的内容还将适当地出现在本书的其他位置上。

www.deitel.com

访问该站点可以查看有关 Deitel & Associates 出版信息的更新、更正及一些相关的资料, 这包括本书中出现的错误、常见问题 (FAQ)、链接以及有关代码及 PowerPoint 幻灯片的下载。

www.prenhall.com/deitel

在 Prentice Hall 网站的这个 Deitel & Associates 页面里包含了有关本书的出版、代码及 PowerPoint 幻灯片下载的信息。

www.softlord.com/comp

访问该网站了解更多的有关计算机的发展历史。

www.elsop.com/wrc/h_comput.htm

该网站上列有数据处理技术的相关历史。讨论了计算机领域中的一些知名人士, 编程语言的演变及操作系统的发展历程。

www.w3.org/History.html

访问该网站可以查看万维网的历史。

www.netvalley.com/intval.html

该网站上谈到了 Internet 的历史。

java.sun.com

Sun Microsystems 公司的这个 Java 网站上面包含了 J2SDK (及其他一些在 Java 开发过程中的较为有用的软件)、代码示例、用于访问 Java API 的文档, 以及指导读者学习 Java 的一些教程的链接。

developer.java.sun.com/developer/onlineTraining/new2java

这个 New to Java Programming Center 的网站上面提供了相关资料的一个集合, 包括文章、教程、用于下载的代码以及测试所学内容的一些试题和难点。

1.12 小结

在本教程里，读者了解到计算机的组织形式。学习了各种不同的编程语言，以及哪些语言（包括Java）是需要翻译器程序的；读者熟悉了一些最流行的编程语言；了解到了结构化编程和面向对象编程的重要地位。还学习了Internet与万维网的一个简短历史和有关Java程序设计语言的发展历史。

我们采用了一个实用的Java应用程序作为一种“探试”。在这个过程中，读者了解到Java提供了许多现有的一些可执行有用功能的GUI组件，通过熟悉这些组件所提供的功能，读者便可开发出功能强大的应用程序，而这要比自己试图去完全地构建它们，从速度上来讲快得多。另外，鼓励读者使用其他相关的信息去浏览有关本书，Internet，Web和Java的一些网站。

在下一个教程里，将学习如何使用一些现有的Java组件——JLabel和JFrame。使用JLabel组件可以在JFrame上显示文本和图片。这将有助于创建属于自己的应用程序。读者在学习的过程中将继续使用应用程序驱动的方法，利用这种方法，将看到一些较为有用的应用程序中所采用的Java特性，并且能够：

1. 研究某个应用程序的用户需求。
2. 探试应用程序一个可使用的版本。
3. 学习用来构建属于自己应用程序的一些技术。
4. 构建自己应用程序的版本。

随着读者对本书学习的不断深入，可能会碰到一些关于Java方面的问题，读者只需将E-mail发送至deitel@deitel.com，我们将立即做出回复。我们诚挚地希望读者去享受Java——世界上最广泛使用的一门编程语言的学习，并让这本书一同陪伴在你的左右。祝君好运！

关键术语

Ada 一种以Ada Lovelace女士的名字命名的编程语言，于20世纪70年代到20世纪80年代初期由美国国防部（DoD）主持开发。

算术及逻辑单元（arithmetic and logic unit, ALU） 属于计算机的“制造”部分。ALU可执行计算和做出判断。

汇编语言 一种使用近似英文的缩写来表达计算机中一些基本操作的编程语言。

属性 对象的某个特征的另一种称呼。

带宽 通信线路上承载信息的容量。

BASIC（Beginner's All-Purpose Symbolic Instruction Code，初学者通用符号指令码） 一门在20世纪60年代中期由Dartmouth学院的两位教授Joho Kemeny和Thomas Kurtz开发的可用来编写简单程序的编程语言，其主要目的是让初学者熟悉一些编程技术。

中央处理单元（central processing unit, CPU） 计算机中用于负责监控计算机其他单元操作的硬件部分。

类 一种用于表示一组相似对象的类型。某种类可指明其对象的通用形式：属性及从属于此类对象的某些可用的操作。对象与其类的关系类似于房子与其蓝图之间的关系。

.class文件 包含了已从Java转换成某种可由Java运行环境（JRE）识别的相应语句形式的一种文件。

COBOL（Common Business Oriented Language，面向商业的通用语言） 是在20世纪50年代后期，由一批同政府及商用计算机用户进行合作的计算机制造商所开发的一门编程语言。该语言主要是在管理大量数据的商用应用程序中使用的。

编译程序 一种可将高级语言程序转换成机器语言的翻译器程序。

计算机 一种在完成相同的任务时，能以超过人类上百万甚至是十亿倍的速度执行计算并做出逻辑判断的设备。

计算机程序 指导计算机经过一个有序操作序列的指令集。

计算机程序员 利用编程语言编写计算机程序的一类人。

数据库 一种组织化的信息集合。

对话框 一种向用户显示一条信息或者显示用户可从中选择不同操作的窗口。

拖拽鼠标 按住鼠标键移动鼠标的操作。

动态内容 一种具备了动画和交互内容的类型。

第一代语言 机器语言的另一个名称。

FORTRAN (FORMula TRANslator, 公式转换语言) 一种由 IBM 公司在 20 世纪 50 年代中期开发的, 专为创建需要有复杂数学计算的科学与工程方面应用程序的编程语言。

第四代语言 (fourth generation language, 4GL) 某种编程语言, 类似于英文这样的自然语言, 主要用来管理数据库中所存储的信息。

框架类库 (Framework Class Library, FCL) 一个专为 Microsoft 的 .NET 平台开发的一个可复用软件组件的强大类库。FCL 提供了与 Java 类库功能相似的功能。

图形用户界面 (graphical user interface, GUI) 应用程序中提供用户交互的可视化部分。

GUI 组件 一种可复用的组件, 如 JButton 或 JComboBox, 通过它可使用户同应用程序进行交互。

硬件 组成计算机的各种不同的设备, 包括键盘、屏幕、鼠标、硬盘驱动器、存储器、CD-ROM 及 DVD 驱动器以及处理单元, 等等。

高级语言 一类编程语言, 其单条语句便能完成相当多的任务。高级语言指令看起来非常类似于日常的英文, 并可包含常用的数学标记。

超文本标记语言 (HyperText Markup Language, HTML) 一种通过超链接文档标记万维网上可用于共享信息的语言。

输入单元 计算机的“接收”部分, 可以从各种不同的输入设备, 如键盘、鼠标等之中获取信息 (数据和计算机程序)。

Internet 一个世界范围内的计算机网络。今天大部分的人们是通过万维网来访问 Internet 的。

解释器 能够直接执行高级语言程序而无需将这些程序编译成机器语言的一种程序。

.java 文件 一种包含 Java 源代码的文件。

Java API (Application Programming Interface, 应用程序编程接口) 作为 Java 编程语言的一部分, 提供了一个现有类的集合。

Java 类库 参照 Java API。

Java 运行环境 (Java Runtime Environment, JRE) J2SDK 中用来执行 (运行) Java 程序的一部分。

Ada Lovelace 女士 一位因其在 19 世纪早期所做的工作, 被公认为是世界上的第一位计算机程序员。

与机器相关 仅存在一种支持某个与特定机器相关的技术的计算机平台。

机器语言 某种计算机的“自然”语言, 通常是由指挥计算机处理最基本操作的数字流 (多个 0 和 1) 所组成的。

存储单元 能够用于快速访问, 而相对来说是低容量的这个计算机的“仓库”部分, 会临时存储某个应用程序运行时的数据。

方法 某个 Java 类中的一部分, 用于执行一个任务并且在任务完成时可能会返回相应的信息。

微处理器 能够使计算机进行工作的芯片 (即计算机的“大脑”)。

对象 一个能模拟现实世界中的一个实体的可复用的软件组件。

对象技术 一种用于创建关注特定应用程序领域中的一些有意义的软件单位的包装模式。对象的例子包括日期对象、时间对象、支票对象和文件对象, 等等。

输出设备 可将计算机处理的信息发送出去的设备。

输出单元 属于计算机的一部分, 它能够取出计算机所处理的信息并将其放置在各种不同的输出设备上, 且这些信息可在计算机的外部使用。

调色板 各种颜色的集合。

Pascal 一种以 17 世纪数学家及哲学家 Blaise Pascal 的名字命名的编程语言。该语言用于讲授结构化的编程思想。

与平台无关 不依赖于某个特定的计算机系统。Java 编程语言就是与平台无关的, 因为 Java 程序可在各种不同的系统中创建和运行。

过程化的编程语言 一种强调操作(动词)而非事物或对象(名词)的编程语言(如 FORTRAN, Pascal, BASIC 和 C)。

特性 对象的一个属性, 如大小、颜色和重量。

第二代语言 汇编语言的另一种名称。

二级存储单元 一个属于计算机中的长效、高容量的“仓库”部分。

软件 在计算机中运行的程序。

软件复用 软件开发的一种方法, 能够使程序员通过复用现有的软件组件快速地开发出新的应用程序。

源代码文件 一种具有扩展名为 .java 且用于存储程序员所编写代码的文件。

结构化编程 具备一定规则的用来创建清晰、正确, 以及易于修改的软件的一种方法。

超级计算机 能够以每秒千亿次的速度执行计算的计算机。

第三代语言 高级语言的另一种名称。

传输控制协议 /Internet 协议 (Transmission Control Protocol/Internet Protocol, TCP/IP) Internet 上通信协议的组合集。

万维网联盟 (World Wide Web Consortium, W3C) 一种给不同的个人和公司提供合作及开发万维网技术的论坛。

万维网 (World Wide Web, WWW) 一个相关 Internet 的硬件和软件的集合, 能够允许计算机用户定位并浏览基于多媒体的文档(将各种文本、图像、动画、音频、视频组合在一起的文档)。

习题

选择题

- 1.1 万维网的开发是 ____。
 - a) 由 ARPA 完成的
 - b) 由 CERN 的 Tim Berners-Lee 完成的
 - c) 在 Internet 之前
 - d) 作为 Internet 的一个替代
- 1.2 拥有扩展名为 ____ 的文件可以存储程序员所编写的 Java 源代码。
 - a) .java
 - b) .class
 - c) .exe
 - d) .jre
- 1.3 JComboBoxes, JButton 和 JPanel 都属于 ____ 的实例。
 - a) 平台
 - b) 高级语言
 - c) 方法
 - d) GUI 组件
- 1.4 ____ 是属于主存的一个实例。
 - a) TCP
 - b) RAM
 - c) ALU
 - d) CD-ROM
- 1.5 Java 是 ____ 语言的一个实例, 其单条程序语句便可完成更多的任务。
 - a) 机器
 - b) 汇编
 - c) 高级
 - d) 以上均不对
- 1.6 主要用来创建“网络中的网络”的协议是 ____。
 - a) TCP
 - b) IP
 - c) OOP
 - d) FCL
- 1.7 ____ 编程语言不属于面向对象语言。
 - a) C
 - b) C++
 - c) Java
 - d) Visual Basic .NET
- 1.8 应用程序中允许用户进行交互的可视化部分为 ____。
 - a) 事件
 - b) 平台
 - c) GUI
 - d) 类库
- 1.9 通信线路上承载信息的容量被称为 ____。
 - a) 网络
 - b) 二级存储器
 - c) 通信量
 - d) 带宽
- 1.10 Java 编程语言提供了可由开发人员使用的 ____, 因而每个应用程序不必从头创建。
 - a) 现有类库
 - b) TCP
 - c) 汇编代码
 - d) 二级存储器

练习题

1.11 将下列项目分别归类为硬件和软件：

a) CPU b) 编译程序 c) 输入单元 d) 一个字处理程序 e) 一个 Java 程序

1.12 翻译器程序，如汇编程序和编译程序，可将程序从一种语言（源语言）转换成另一种语言（目标语言）。判断下列语句中哪些为真，哪些为假（若为假，请解释原因）：

a) 编译程序将高级语言程序翻译成目标语言程序。

b) 汇编程序将源语言程序翻译成机器语言程序。

c) 编译程序将源语言程序翻译成目标语言程序。

d) 高级语言通常是与机器相关的。

e) 机器语言程序在某台计算机上运行之前需进行翻译。

1.13 计算机可以被分解为 6 个单元。

a) 哪一个单元可以认为是其他单元的“领导”？

b) 哪一个单元拥有高容量的“仓库”并能够在计算机关闭以后仍旧保留信息？

c) 哪一个单元可判断内存中所存储的两个项目是否是相等的？

d) 哪一单元能够从键盘和鼠标这样的设备中获取信息？

1.14 写出以下首字母缩略词的全称：

a) W3C

b) TCP/IP

c) OOP

d) JRE

e) HTML

1.15 面向对象编程技术的优点是什么？

教程 2 Welcome 应用程序

引入图形用户界面的程序设计



教学目标

在本教程中，读者将学到以下内容：

- 设置 JFrame 标题栏文本
- 改变 JFrame 的背景色
- 将 JLabel 放置在 JFrame 上
- 显示 JLabel 组件中的文本
- 显示 JLabel 组件中的图片
- 执行应用程序

今天，用户更喜欢使用如点击按钮、输入数据等具有交互式操作的图形用户界面（GUI）软件。所以，现在绝大多数的应用程序都是基于GUI的。利用Java，可以创建出具有各种不同输入及输出信息方式的图形应用程序。所有这些内容，读者都将从本书中学到。

在本教程中，将学习使用GUI编程技术创建一个Welcome应用程序。通过将两个组件（一个是包含文本的JLabel，另一个是包含图片的JLabel）放置在窗口中构建出该应用程序的GUI，并通过设定一些属性自定义出该窗口与JLabel的外观显示。读者还将设置许多属性值，包括：窗口背景颜色的设置、设置JLabel中的文本（"Welcome to Java Programming!"）、设置JLabel中的图片等内容。

2.1 探试 Welcome 应用程序

在上一教程里，我们已介绍过一些有关Java的知识。在本教程中，读者将使用Java技术构建一个Welcome应用程序。这个应用程序必须满足下面的需求：

应用程序需求分析

某个软件公司(Deitel & Associates)要求开发一个包含问候语“Welcome to Java Programming!”和一幅该公司虫形吉祥物图片的简单Welcome应用程序。

读者将以这个已经完成的应用程序的探试作为起点。之后，学习另外一些Java技术，作为创建属于自己应用程序的基础。



探试 Welcome 应用程序

1. 定位到完成后的应用程序 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\Examples\Tutorial02\CompletedApplication\Welcome` 然后按下 Enter 键（如图 2.1 所示），将目录定位到这个完成后的 Welcome 应用程序目录之下。



图 2.1 定位到完成后的 Welcome 应用程序中

2. 运行 Welcome 应用程序 在命令提示符窗口下键入 `java Welcome` 并按下 Enter 键运行该应用程序 (如图 2.2 所示)。记住 Java 命令是区分大小写的, 因此 Welcome 必须要有大写, 否则此应用程序将不予执行。图 2.3 显示了该应用程序的执行结果。



图 2.2 运行完成后的 Welcome 应用程序

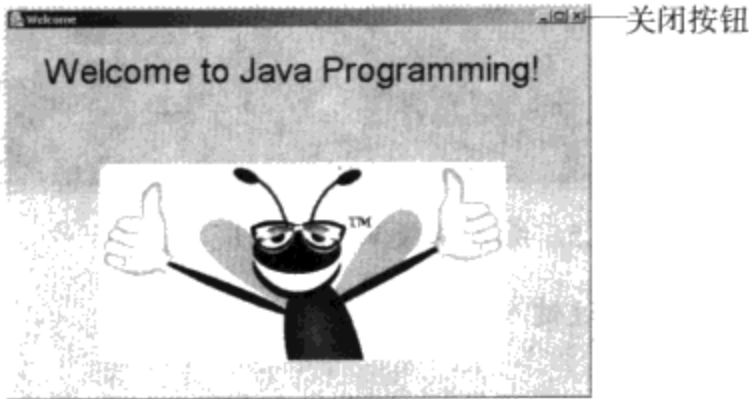


图 2.3 Welcome 应用程序的执行结果

- 3. 关闭运行的应用程序 点击关闭按钮 关闭正在运行的应用程序。返回到命令提示符窗口下。
- 4. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

2.2 编译并运行模板 Welcome 应用程序

在教程 1 中, 读者已经知道了 Java 运行环境 (JRE) 只可以运行具备 .class 扩展名的文件。而 .class 文件的产生只能是通过编译包含应用程序代码的 Java 源代码文件 (具有 .java 扩展名的文件) 才可以得到。接下来, 我们将学习如何编译 Welcome 应用程序。

编译 Welcome 应用程序

- 1. 将模板复制到工作目录中 将 `C:\Examples\Tutorial02\TemplateApplication\Welcome` 目录复制到 `C:\SimplyJava` 目录中。
- 2. 定位到模板应用程序中 选择 `Start → Programs → Accessories → Command Prompt` 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\Welcome` 然后按下 Enter 键 (如图 2.1 所示), 将目录改变到当前的工作目录 Welcome 下。键入 `dir` 并按下 Enter 键列出此目录下的全部内容 (如图 2.4 所示)。目录中应该存在一个 `Welcome.java` 文件, 但并不包含 `Welcome.class` 文件。
- 3. 编译应用程序 在命令提示符窗口中, 键入:
`javac Welcome.java`
并按下 Enter 键。javac 命令将编译 Java 源代码文件并在该目录下产生一个 `Welcome.class` 文件。再次键入 `dir` 并按下 Enter 键, 这次看到目录中包含了一个 `Welcome.class` 文件 (如图 2.5 所示)。



图 2.4 定位到模板 Welcome 应用程序



图 2.5 编译模板 Welcome 应用程序

4. 一旦创建 .class 文件，就可以通过键入：

```
java Welcome
```

并按下 Enter 键来运行这个应用程序了。这时，应用程序是一个空白的 JFrame（如图 2.6 所示），即一个空白的窗口。需要读者创建的每一个应用程序都会显示在其自身的 JFrame 中。我们通常将其称为“窗口”或者是“应用程序窗口”。本教程的余下部分，将自定义两个在 JFrame 中显示的 JLabel 组件，如图 2.3 所示。

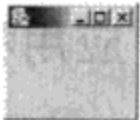


图 2.6 运行模板 Welcome 应用程序

- 5. 关闭运行的应用程序 点击关闭按钮 关闭正在运行的应用程序。返回到命令提示符窗口下。
- 6. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

自测题

- 1. 一个源代码文件应具备 _____ 扩展名。
a) .class b) .source c) .java d) 以上均不是
- 2. _____ 可将一个 .java 文件转换成一个 .class 文件。
a) 调试程序 b) 编译程序 c) 转换器程序 d) JRE

答案：1) c 2) b

2.3 创建 Welcome 应用程序

本节中，读者将开发一个属于自己的 Welcome 应用程序。该应用程序是由一个使用两个 JLabel 组件的 JFrame 构成的。JLabel 组件可用于显示任何不允许用户进行修改的文本或者是一幅图片。回想我们曾在 1.4 节和 1.7 节讨论过，JLabel 这个名字实际代表的是 Java 中的一个类。本例中，读者将创建的两个 JLabel 组件便属于该 JLabel 类的两个对象。下面，开始创建 Welcome 应用程序。

改变 JFrame 标题栏中的文本

1. 打开 Welcome 应用程序的模板文件 定位于存储该应用程序所在的 C:\SimplyJava\Welcome 目录。找到 Welcome.java 文件并用自己的文本编辑器或者集成开发环境 (IDE) 将其打开。尽管每种文本编辑器和 IDE 是不同的, 但结果都应该与图 2.7 中所显示的代码类似 (注意: 当使用文本编辑器查看应用程序的代码时, 切记所看到的只是应用程序代码的一部分。同样, 为了帮助读者更好地完成程序的编写, 我们将在前面几个教程中提供所需要的大多数代码, 并只要求读者能在每个应用程序的代码中插入或者修改其中的一部分)。

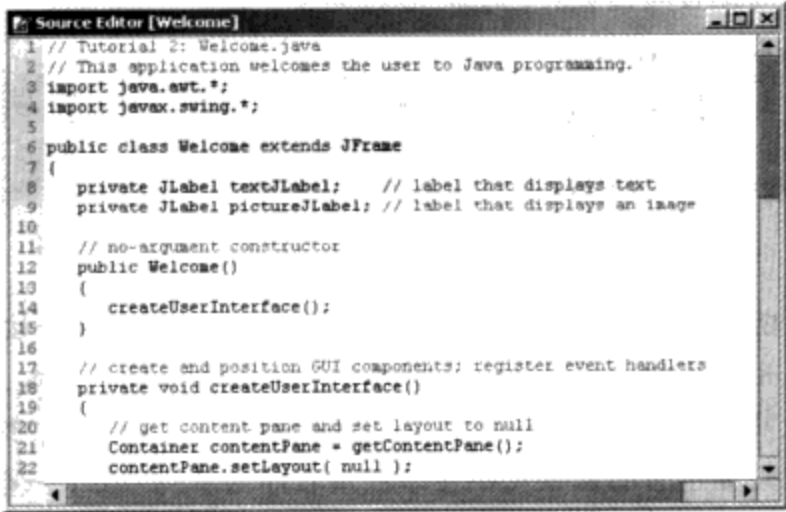


图 2.7 通过一个编辑器打开的 Welcome 应用程序的 Java 源代码 (注意: 所有用于显示类似本图源代码的窗口是通过 Sun ONE Studio 4 Community Edition 创建的。
©Copyright 2003 Sun Microsystems, Inc. All rights reserved. Used by permission)

2. 设置 JFrame 标题栏的文本 将图 2.8 中的第 33 行插入到应用程序中。此行可将应用程序的标题属性设置为 "Welcome"。标题属性用于显示 JFrame 标题栏 (包含窗口标题、属于窗口的顶部区域; 参见图 2.3 中顶部的单词 Welcome) 中的文本。JFrame 的标题名应该是简短的、类似书名大写形式 (每个重要单词的首字母应为大写, 不应使用任何标点符号作为结束) 且具备描述性质的名字 (例如, Capitalization in a Book Title)。

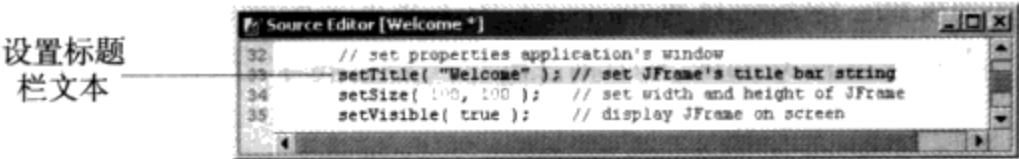


图 2.8 设置 JFrame 标题栏的文本



GUI 设计提示

选择简短的具备描述性质的 JFrame 标题。

看一下这条插入的代码:

setTitle("Welcome");

这是一条用于执行某项操作的 Java 语句。语句都是以分号 (;) 作为结束的。而此条语句将把 JFrame 的标题属性值设置为 "Welcome"。

双引号中封闭的文本被称为字符串或者叫字符串文字。这里, 字符串表示的是希望在 JFrame 的标题栏中显示的文本。字符串同样也是 Java 中的一个类。

Java 中的代码是区分大小写的。这意味着大写和小写的单词会受到不同的对待。例如, setTitle 和 setTitle 被认为是不同的标识符。



GUI 设计提示

JFrame 标题应使用书名大写形式,即每个重要单词的首字母应为大写,且不应使用任何标点符号作为结束。

- 3. 保存应用程序 保存修改后的源代码文件。在大多数编辑器中,可以通过选择 File → Save 完成保存。
- 4. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\Welcome` (如图 2.9 所示) 并按下 Enter 键进入到 Welcome 工作目录中。
- 5. 编译应用程序 通过键入命令 `javac Welcome.java` (如图 2.9 所示) 并按下 Enter 键,对该应用程序进行编译。应用程序若不能正确编译,则命令提示符窗口中会显示错误信息。假若这个应用程序的确不能够正确编译,请检查是否与图 2.8 中的第 33 行完全一致。



图 2.9 对更新后的 Welcome 应用程序进行编译

- 6. 运行应用程序 若此应用程序能够正确编译,通过键入 `java Welcome` 并按下 Enter 键来运行它。图 2.10 显示的是更新后的应用程序的运行结果。注意标题栏中现在出现了文本,但却没有完全显示出来,这是因为这个 JFrame 太小了。用户可通过拖拽四个角来调整窗口的大小。然而,最好是在应用程序开始运行的时候就已通过设置适当的属性大小显示应用程序的窗口。在下一个框图中,将对 JFrame 的大小进行改变。



图 2.10 设置了标题的 JFrame

- 7. 关闭应用程序 点击关闭按钮 关闭正在运行的应用程序。返回到命令提示符窗口下。
- 8. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

可以通过设置 size 属性调整 JFrame 的大小,该属性是通过称为像素(图片的元素)的单位大小来指定 JFrame 的宽度和高度的。像素是计算机屏幕上用于显示某种颜色的一个微小的点。注意, JFrame 的高度包含标题栏。读者在设置 JFrame 大小属性的框图中,将学习如何设置 JFrame 的宽度与高度。

设置 JFrame 的大小属性

- 1. 设置 JFrame 的宽度和高度 对应用程序中的 setSize 语句进行修改(第 34 行),如图 2.11 所示。第一项参数值(608)代表的是以像素为单位的 JFrame 的宽度,而第二项值(413)代表的是以像素为单位的 JFrame 的高度。读者应该注意到该语句与前一个设定某属性的那个示例在形式上是一致的。都需要写上 set, 然后跟一个以大写字母为起始的属性名(即 Size),接下来是由一对括号封闭起来的属性值,最后加上一个分号。

改变JFrame的
宽度和高度

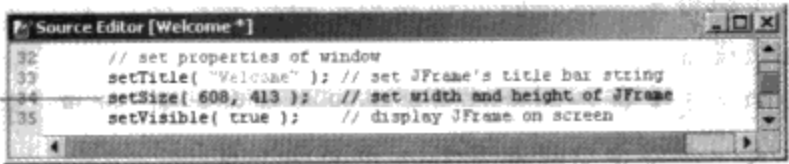


图 2.11 大小属性控制 JFrame 的宽度和高度

- 2. 保存应用程序 保存修改后的源代码文件。在大多数编辑器中,可以通过选择 File → Save 完成保存。

- 3. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\Welcome` (如图 2.12 所示) 并按下 Enter 键进入到 Welcome 工作目录中。
- 4. 编译应用程序 通过键入命令 `javac Welcome.java` (如图 2.12 所示) 并按下 Enter 键, 对该应用程序进行编译。假若这个应用程序不能正确编译, 请检查与图 2.11 中的第 34 行是否完全一致。



图 2.12 对更新后的 Welcome 应用程序进行编译

- 5. 运行应用程序 若此应用程序能够正确编译, 可以通过键入 `java Welcome` 并按下 Enter 键来运行它。图 2.13 显示的是更新后的应用程序的运行结果。注意, 现在可在标题栏中看到完整的标题了, 但 JFrame 里面仍旧是空的。

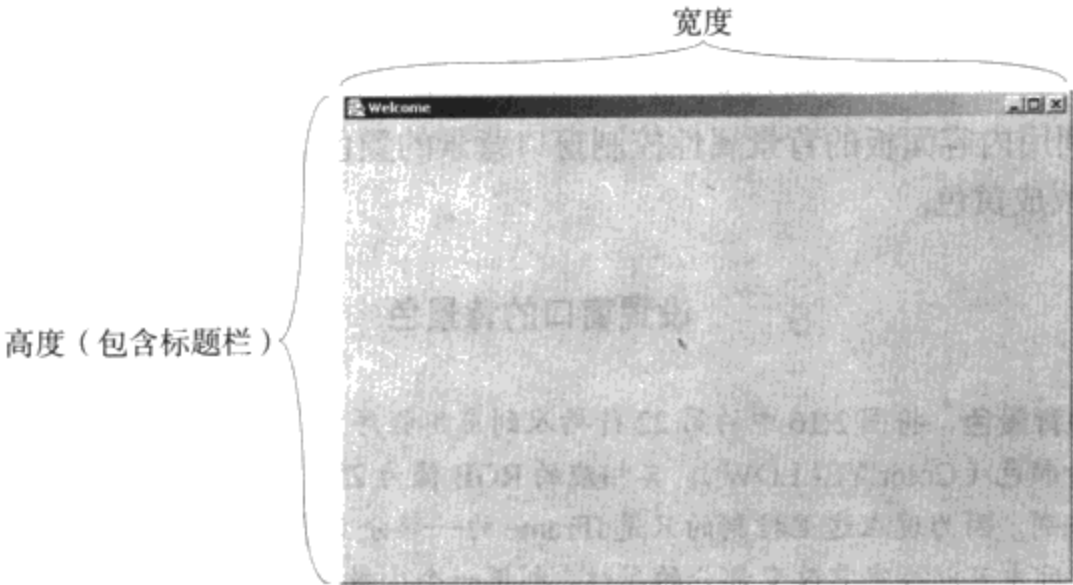


图 2.13 设定了标题与大小属性的 JFrame

- 6. 关闭应用程序 点击关闭按钮 关闭正在运行的应用程序, 返回到命令提示符窗口下。
- 7. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

既然 JFrame 的大小已经设定, 下面将进一步通过对其背景色的改变 (从灰色变为黄色) 来自定义这个应用程序窗口。屏幕上的每种颜色是通过组合红、绿、蓝 3 个数值进行创建的。它们一起被称为 RGB 值。三个 RGB 值均是从 0 到 255 范围内的整数值。第一个 RGB 值指的是红色的数量值, 第二个值指的是绿色的数量值, 而第三个则为蓝色的数量值。因而, 0, 0, 0 表示黑色; 255, 255, 255 为白色; 而 255, 0, 0 则为红色。Java 为我们提供了 13 种预定义的颜色, 参见图 2.14 中所列的各项。后面, 读者将学习如何创建用于指定颜色的 Color 对象。此刻, 值得瞩目的是, 我们可以尝试着使用这些总共可达 1670 万种的颜色。

常量	RGB值	常量	RGB值
Color.BLACK	0, 0, 0	Color.MAGENTA	255, 0, 255
Color.BLUE	0, 0, 255	Color.ORANGE	255, 200, 0
Color.CYAN	0, 255, 255	Color.PINK	255, 175, 175
Color.DARK_GRAY	64, 64, 64	Color.RED	255, 0, 0
Color.GRAY	128, 128, 128	Color.WHITE	255, 255, 255
Color.GREEN	0, 255, 0	Color.YELLOW	255, 255, 0
Color.LIGHT_GRAY	192,192,192		

图 2.14 预定义的颜色及相对应的 RGB 值

每个 `JFrame` 都有一个称为内容面板的容器对象。此容器的作用是提供一个用于放置如 `JLabel` 这样的 GUI 组件的可视化区域。内容面板占据着 `JFrame` 边界内完整的可视化区域。也就是从 `JFrame` 标题栏以下直到 `JFrame` 最低端的边界处（如图 2.15 所示）。

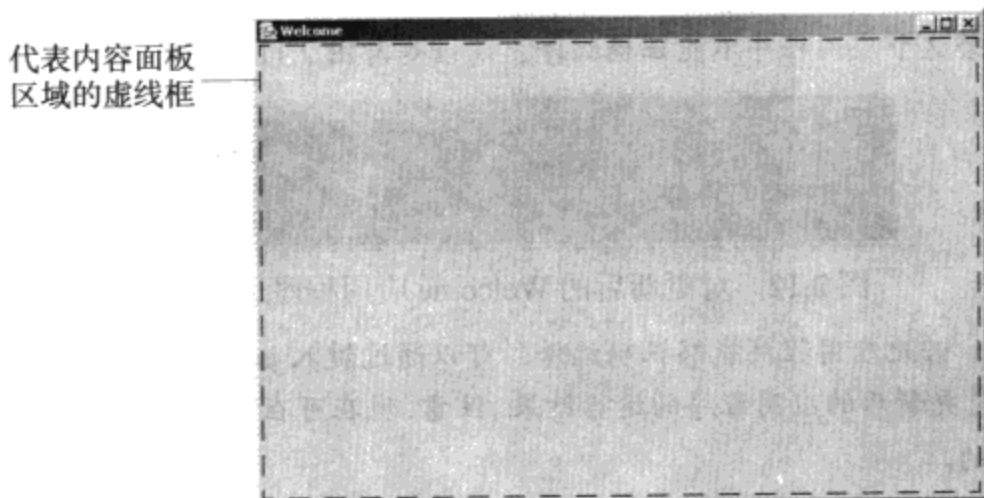


图 2.15 突出显示内容面板的 `JFrame`

接下来，将利用内容面板的背景属性控制窗口背景的颜色。下面，读者将学习如何把内容面板的背景色从灰色改成黄色。

设置窗口的背景色

1. 改变窗口的背景色 将图 2.16 中的第 22 行插入到应用程序中。此行将把内容面板的背景属性设定为预定义的黄颜色（`Color.YELLOW`）。其相应的 RGB 值为 255, 255, 0。这行代码同上面几行的格式多少有一些差别。因为现在这里控制的只是 `JFrame` 的一部分（其内容面板），因此必须首先指出是哪一部分，然后才可以指定需改变部分的属性。利用一个小数点将这一部分同属性分隔开。这种表示法将在教程 4 中完整解释。

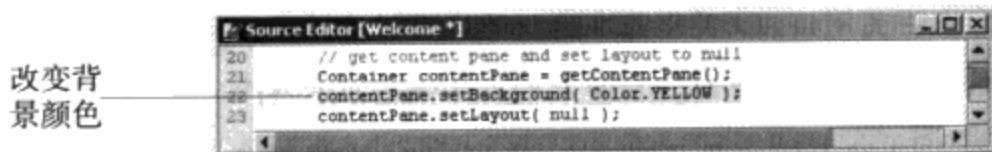


图 2.16 将 `JFrame` 的背景属性设定为黄色



GUI 设计提示

可以在应用程序中使用不同的颜色，但不应超出分散用户注意力的程度。

2. 保存应用程序 保存修改后的源代码文件。
3. 打开命令提示符窗口改变目录 选择 `Start` → `Programs` → `Accessories` → `Command Prompt` 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\Welcome`（如图 2.17 所示）并按下 `Enter` 键进入到 `Welcome` 工作目录中。
4. 编译应用程序 通过键入命令 `javac Welcome.java`（如图 2.17 所示）并按下 `Enter` 键，对该应用程序进行编译。假若这个应用程序不能够正确编译，请检查与图 2.16 中的第 22 行是否完全一致。



图 2.17 对更新后的 `Welcome` 应用程序进行编译

5. 运行应用程序 若此应用程序能够正确编译，可以通过键入 `java Welcome` 并按下 `Enter` 键来运行它。图 2.18 显示的是更新后的应用程序的运行结果。

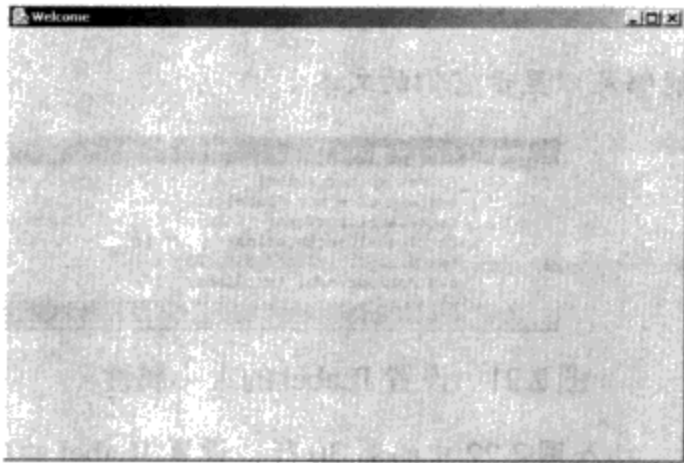


图 2.18 设定了标题、大小和背景属性的 JFrame

- 6. 关闭应用程序 点击关闭按钮 关闭正在运行的应用程序。返回到命令提示符窗口下。
- 7. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

现在，读者已完成了 JFrame 及其内容面板的自定义，下面将自定义一个在 JFrame 中用于显示一条文本问候语的 JLabel 组件。

在内容面板中自定义一个 JLabel

- 1. 自定义 JLabel 的外观 将图 2.19 中的第 27 行插入到这个应用程序的代码中。此行用于将 JLabel 的文本属性设置为 "Welcome to Java Programming!"。当显示某个 JLabel 时，其文本属性将显示在 JFrame 中的这个 JLabel 上。

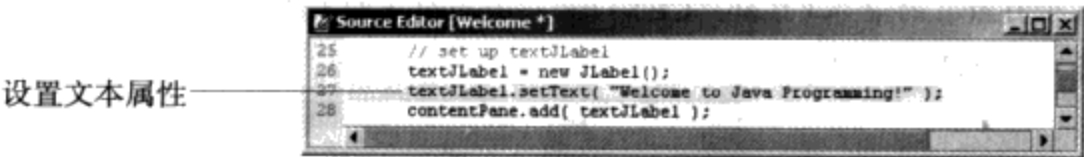


图 2.19 设置 JLabel 的文本属性



GUI 设计提示
使用 JLabel 显示不允许用户进行修改的文本。

- 2. 设置位置属性 插入图 2.20 中的第 28 行以设置 JLabel 的位置属性。

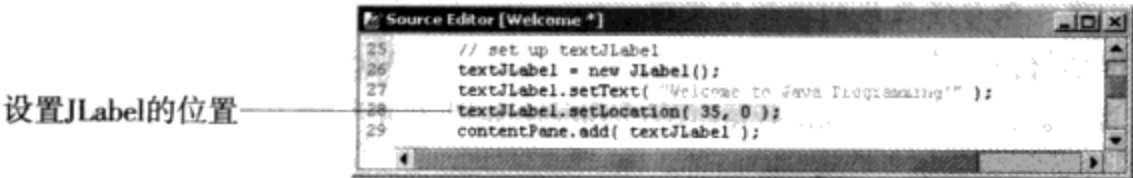


图 2.20 设置 JLabel 的位置属性

JLabel 的位置属性指定了该组件的左上角在内容面板中的位置。Java 是将 0, 0 值分配给了内容面板（位于窗口标题栏之下）的左上角位置处。组件位置属性的设置是根据其与内容面板上 0, 0 点的距离多少来进行计算的。当位置属性的第一个值增加时，组件将沿 x 轴方向向右移动。当位置属性的第二个值增加时，组件将沿 y 轴方向向下移动。本例中，作为 35, 0 的值表示此 JLabel 将放置在距离内容面板左上角向右 35 个像素点及距离内容面板左上角向下 0 个像素点的位置上。而作为 35, 48 的值则表示此 JLabel 将放置在距离内容面板左上角向右 35 个像素点及距离内容面板左上角向下 48 个像素点的位置上。

- 3. 设置大小属性 插入图 2.21 中的第 29 行，设置 JLabel 的大小属性。JLabel 的大小属性表示的是以像素为单位的 JLabel 的宽度和高度，这与 JFrame 的大小属性是一样的。事实上，大多数 Java 组件都拥有一些相同的属性。这使得组件更容易学习和使用。



GUI 设计提示

保证所有的 JLabel 组件能够足以显示它们的文本。

设置JLabel的大小

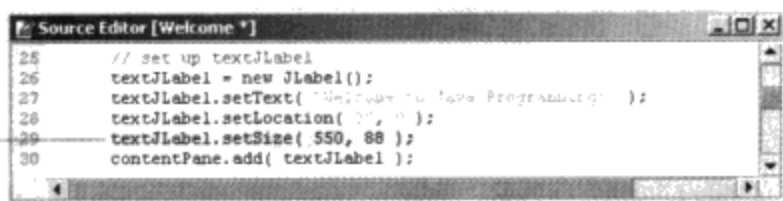


图 2.21 设置 JLabel 的大小属性

4. 设置 JLabel 的字体属性 插入图 2.22 中的第 30 行，设置 JLabel 的字体名称、字形及字号。设置某个 JLabel 的字体属性包括设置字体的名称（Times，Courier 等）、字形（常规、粗体、斜体等）以及按照点数（一个点等于 1/72 英寸^①）的字号（16，18 等）。本条语句是将字体名称设置为 "SansSerif"，字形设置为 Font.PLAIN，字号设置为 36。名为 "SansSerif" 的字体名实际描述的是包含 Arial 和 Helvetica 的一组相联系的字体。如果文本比这一 JLabel 还大，那么文本将会被截取。

设置JLabel的字体

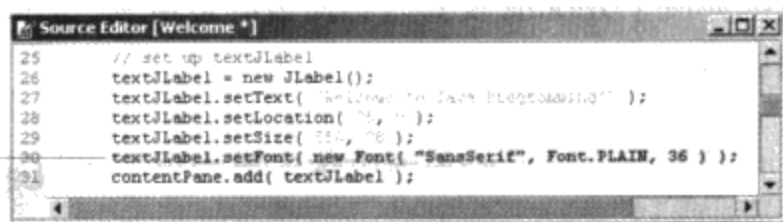


图 2.22 将 JLabel 的字体大小改为 36

5. 对齐 JLabel 中的文本 为了对齐 JLabel 中的文本，需要设定 JLabel 的 horizontalAlignment 属性。该属性允许将 JLabel 中的文本向右、居中或者是向左对齐。将图 2.23 中的第 31 行插入到应用程序中。此文本现在位于 JLabel 的中部 (JLabel.CENTER)。JLabel 中的文本还可以靠左 (JLabel.LEFT，默认值) 或靠右对齐 (JLabel.RIGHT)。

JLabel中居中的文本

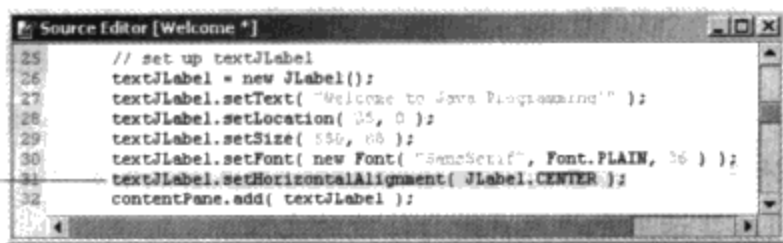


图 2.23 JLabel 中居中的文本

6. 保存应用程序 保存修改后的源代码文件。
7. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\Welcome（如图 2.24 所示）并按下 Enter 键进入到 Welcome 工作目录中。
8. 编译应用程序 通过键入命令 javac Welcome.java（如图 2.24 所示）并按下 Enter 键，对该应用程序进行编译。假若这个应用程序不能够正确编译，请检查是否与图 2.23 中的第 27 行至第 31 行完全一致。



图 2.24 对更新后的 Welcome 应用程序进行编译

9. 运行应用程序 若此应用程序能够正确编译，可以通过键入 java Welcome 并按下 Enter 键来运行它。图 2.25 显示的是更新后的应用程序的运行结果。

^① 1 英寸 = 2.54 cm ——编者注。

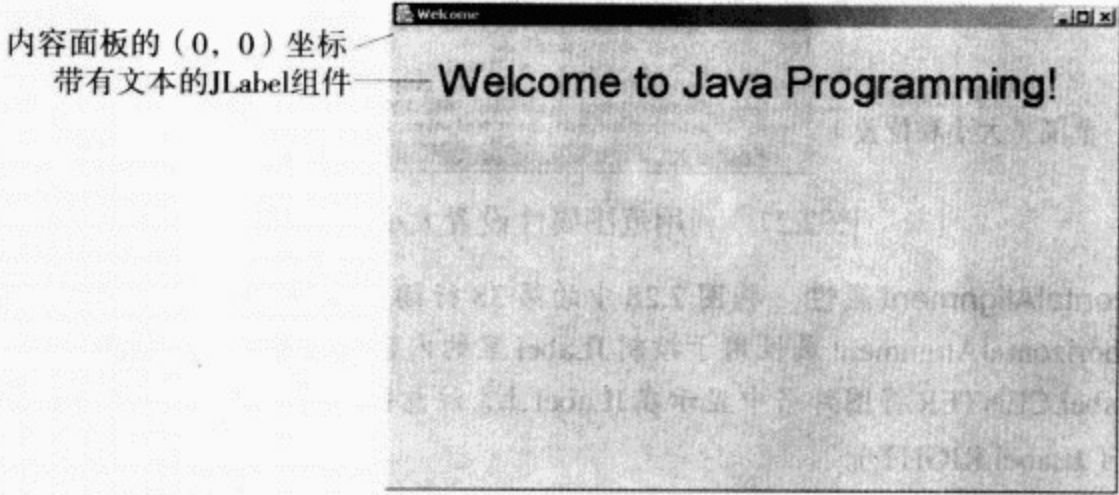


图 2.25 修改 JLabel 文本后的 Welcome 应用程序

- 10. 关闭应用程序 点击关闭按钮 关闭正在运行的应用程序。返回到命令提示符窗口下。
- 11. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

在下面“插入图片并运行Welcome应用程序”中,将指导读者自定义一个用于显示图片的JLabel组件并以此作为整个应用程序的结束。

插入图片并运行 Welcome 应用程序

- 1. 设置 JLabel 的图标属性 当 JLabel 的图标属性设置好以后,会有一幅图片显示在 JLabel 中。将图 2.26 中的第 36 行插入到应用程序中。此行会使文件 "bug.png" 中的图片显示在 JLabel 中。除非这条语句中的字符串指定了包含图片的完整路径 (如 "C:\SimplyJava\Welcome\bug.png"), 否则该条语句会假定图片与应用程序是处在同一个目录下的 (正如本例)。

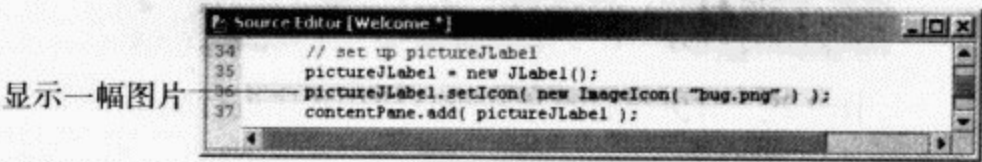


图 2.26 通过设置图标属性在 JLabel 上显示一幅图片



GUI 设计提示

利用带有图片的 JLabel 可以改善那些不允许用户改动图形的 GUI。

常见可用于显示的图片格式有:

- PNG (Portable Network Graphics, 可移植的网络图像文件格式)
- GIF (Graphics Interchange Format, 可交换的图像文件格式)
- JPEG (Joint Photographic Experts Group, 联合图像专家组格式)
- BMP (Windows Bitmap, 位图文件格式)

本应用程序使用的是一幅 PNG 格式的图片。使用图片编辑软件,如 Jasc Paint Shop Pro(www.jasc.com), Adobe Photoshop(www.adobe.com), Microsoft Picture It! (photos.msn.com) 或者是 Microsoft Paint(由 Windows 提供) 可以创建新的图片。本书并不要求读者创建图片;我们将提供所有读者需要的图片。

- 2. 设置 JLabel 的范围属性 插入图 2.27 中的第 37 行。此行用于设置 JLabel 的范围属性。范围属性可控制 JLabel 的大小和位置。前两个参数是将组件的左上角设定在位置 (54, 120) 上。后两个参数则是分别设定组件的宽度与高度 (500, 250)。



GUI 设计提示

保证所有的 JLabel 组件能够完整显示它们的图片。

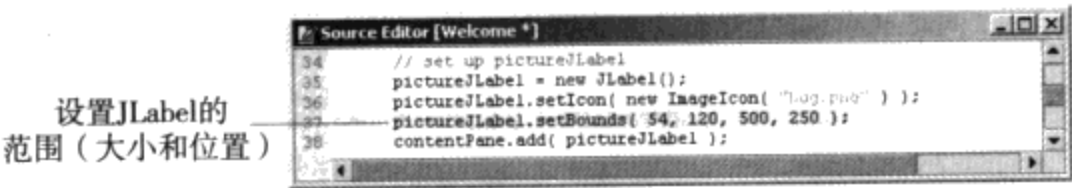


图 2.27 利用范围属性设置大小和位置

3. 设置 `horizontalAlignment` 属性 将图 2.28 中的第 38 行插入到应用程序中，以居中 JLabel 里的图片。JLabel 的 `horizontalAlignment` 属性用于控制 JLabel 里的内容的对齐方式。本例中，我们使用预先定义的常量 `JLabel.CENTER` 将图片居中显示在 JLabel 上。注意图片还可以靠左（`JLabel.LEFT`，默认值）或靠右对齐（`JLabel.RIGHT`）。

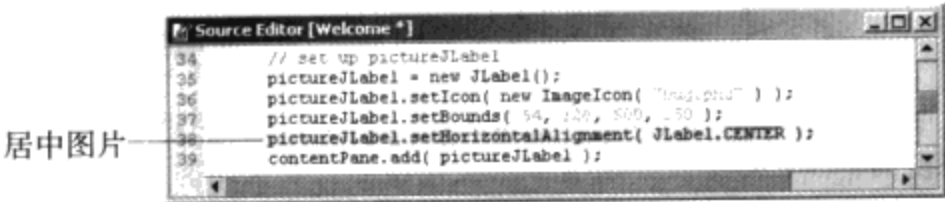


图 2.28 设置 JLabel 的 `horizontalAlignment` 属性

4. 保存应用程序 保存修改后的源代码文件。
5. 打开命令提示符窗口改变目录 选择 `Start → Programs → Accessories → Command Prompt` 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\Welcome`（如图 2.29 所示）并按下 `Enter` 键进入到 `Welcome` 工作目录中。
6. 编译应用程序 通过键入命令 `javac Welcome.java`（如图 2.29 所示）并按下 `Enter` 键，对该应用程序进行编译。假若这个应用程序不能正确编译，请检查是否与图 2.28 中的第 36 行至第 38 行完全一致。



图 2.29 对更新后的 Welcome 应用程序进行编译

7. 运行应用程序 若此应用程序能正确编译，可以通过键入 `java Welcome` 并按下 `Enter` 键来运行它。图 2.30 显示的是完成后的应用程序的运行结果。

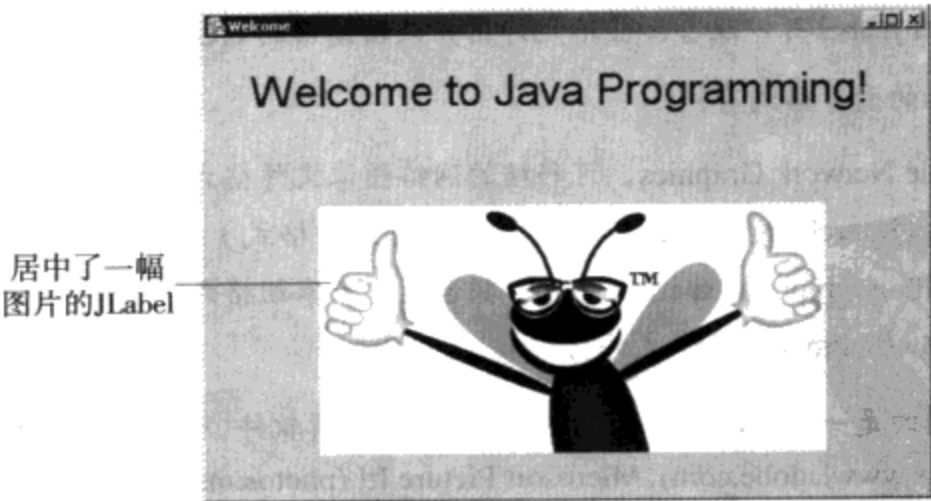


图 2.30 运行完成后的 Welcome 应用程序


8. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。返回到命令提示符窗口下。
9. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

图 2.31 列出了 Welcome 应用程序的源代码。图中，凡在本教程中需要添加、查看及修改的代码行均做出了突出显示。


```

1 // Tutorial 2: Welcome.java
2 // This application welcomes the user to Java programming.
3 import java.awt.*;
4 import javax.swing.*;
5
6 public class Welcome extends JFrame
7 {
8     private JLabel textJLabel;    // JLabel that displays text
9     private JLabel pictureJLabel; // JLabel that displays an image
10
11     // no-argument constructor
12     public Welcome()
13     {
14         createUserInterface();
15     }
16
17     // create and position GUI components; register event handlers
18     private void createUserInterface()
19     {
20         // get content pane and set layout to null
21         Container contentPane = getContentPane();
22         contentPane.setBackground( Color.YELLOW );    将内容面板的背景色设置为黄色
23         contentPane.setLayout( null );
24
25         // set up textJLabel
26         textJLabel = new JLabel();
27         textJLabel.setText( "Welcome to Java Programming!" );
28         textJLabel.setLocation( 35 , 0 );
29         textJLabel.setSize( 550 , 88 );
30         textJLabel.setFont( new Font( "SanSerif", Font.PLAIN, 36 ) );
31         textJLabel.setHorizontalAlignment( JLabel.CENTER );
32         contentPane.add( textJLabel );
33
34         // set up pictureJLabel
35         pictureJLabel = new JLabel();
36         pictureJLabel.setIcon( new ImageIcon( "bug.png" ) );
37         pictureJLabel.setBounds( 54, 120, 500, 250 );
38         pictureJLabel.setHorizontalAlignment( JLabel.CENTER );
39         contentPane.add( pictureJLabel );
40
41         // set properties of application's window
42         setTitle( "Welcome" ); // set JFrame's title bar string
43         setSize( 608 , 413 ); // set width and height of JFrame
44         setVisible( true );    // display JFrame on screen
45
46     } // end method createUserInterface
47
48     // main method
49     public static void main( String[] args )
50     {
51         Welcome application = new Welcome();
52         application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
53
54     } // end method main
55
56 } // end class Welcome

```

图 2.31 Welcome 应用程序的代码

自测题

1. 可以使用 _____ 设置 JFrame 标题栏中出现的文本。
a) setText b) setTitle c) setTitleBar d) setName
2. 可以使用 _____ 指定 JLabel 中显示的图片。
a) setIcon b) setPicture c) setImage d) setGraphic

答案: 1) b 2) a

2.4 语法错误

本教程中, 读者学习了在命令提示符窗口下如何通过输入 `javac Welcome.java` 来编译 Welcome 应用程序。如果不能正确编写代码, 应用程序将不予编译并且命令提示符窗口下会出现相关的错误信息。即便是一个应用程序能够正确编译, 它仍有可能包含错误。调试指的就是在一个应用程序中改正错误的过程。有两种类型的错误——语法错误和逻辑错误。

语法错误(也称为编译错误或者编译时错误)是当由代码所组成的语句违反了某种编程语言的语法规则时出现的错误。这样的错误包括: 当读者在本教程中添加程序代码时, 将 Java 的某个特定的单词拼写错误或者未在每条语句的末尾添加一个分号等。一个应用程序只有在其语法错误更正以后才可以执行, 即只有当它能够正确编译, 才能够执行。

逻辑错误不能阻止应用程序的成功编译, 但它确会使应用程序在运行时产生错误的结果。Java 2 软件开发工具包提供了一个称为调试程序的软件, 该程序允许开发人员对其应用程序进行分析以便查找到相应的逻辑错误。

在编译 Java 应用程序的时候, 任何语法错误都会出现在命令提示符窗口中, 且每一个错误都会配有的一个简短的描述。图 2.32 显示了在图 2.31 中第 37 行的末尾缺少一个分号字符时, 所出现的一个错误信息。这时, 就应该添加分号并重新进行编译。



图 2.32 列有语法错误的命令提示符窗口

在本书一些教程的末尾处, 我们提供了使用调试程序一节, 作为读者学习如何通过 Java 调试程序检查并删除逻辑错误的参考。在下面“使用调试程序: 语法错误”中, 读者将试着生成一些语法错误, 查看编译程序提供的这些错误信息然后对这些错误进行改正。在使用调试程序一节的后半部分, 读者将实际使用这个调试程序。



使用调试程序: 语法错误

1. 打开应用程序 如果 Welcome 应用程序的源代码当前没有打开, 找到 Welcome.java 文件并用文本编辑器或者 IDE 将其打开。
2. 生成语法错误 为进行调试, 读者现在可以自行创建一些语法错误。例如, 在第 27 行的 textJLabel 的末尾添加一个字母 s, 再将第 28 行语句末的右括号删除。图 2.33 显示了修改后(错误的)的两行代码。
3. 保存应用程序 保存修改后的源代码文件。
4. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\Welcome` 并按下 Enter 键进入到 Welcome 工作目录中。

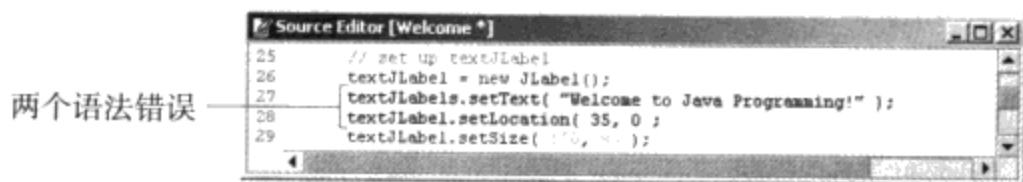


图 2.33 在代码中引入两个语法错误

5. 编译应用程序 通过键入命令 `javac Welcome.java` 并按下 Enter 键，对该应用程序进行编译。图 2.34 显示了由编译程序生成的错误信息。
6. 查找错误信息 编译程序在找到每一个错误后，所显示的信息中会包含相应的文件名（Welcome.java）、错误产生的行号、有关错误的一个简短描述，以及在实际源代码中产生错误的行号。本例中，编译程序通知程序员第 27 行和第 28 行发现了错误。



图 2.34 由编译程序生成的相关语法错误的两条错误信息

7. 改正语法错误 既然编译程序已指出产生语法错误的地方，便可返回源代码并改正在第 2 步中生成的两个错误。保存该文件并再次返回到命令提示符窗口中。重新编译应用程序，此应用程序现在应该能够正确地编译了。
8. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭（注意：不需要关闭应用程序，因为现在并不要求执行它。读者进行的这个编译只是用于测试能否发现语法错误）。

自测题

1. 在查找到应用程序的一个语法错误以后，编译程序将给用户通告此错误_____。
a) 出现的行号 b) 可用于改正错误的正确代码 c) 一个简短的描述 d) 选择 a 和 c
2. 语法错误可因多种原因产生，例如，当_____。
a) 应用程序突然终止 b) 缺少括号 c) 单词拼写错误 d) 选择 b 和 c

答案：1) d 2) d

2.5 小结

本教程引入 Java 的编程技术。读者学习了如何通过改变各种不同 Java 组件的属性来自定义应用程序的图形用户界面。

在创建自己的 Welcome 应用程序时，设置了 JFrame 的标题栏文本和大小（宽度和高度），并且利用背景属性对 JFrame 内容面板的背景色进行了设置。读者也了解到 JLabel 是一种能够显示文本和图片的组件，并通过设置 JLabel 的文本、字体和 `horizontalAlignment` 属性对一个 JLabel 中的文本做出了显示，另外，还通过设置另一个 JLabel 的图标和 `horizontalAlignment` 属性显示了一幅图片。在对这些 JLabel 设置大小和位置属性，或者是设置范围属性以后，可将它们布置在 JFrame 上。

在下一个教程中，将继续学习如何去创建图形用户界面。尤其是，学会创建一个用于接受用户输入的 GUI 组件的应用程序。

技术小结

快速及有效地创建 GUI

- 使用预定义的图形用户界面组件（如 JLabel）。

设置组件的大小和位置

- 使用 setSize 和 setLocation 设置组件的大小和位置属性，或者是通过 setBounds 设置组件的范围属性（该属性将设置组件的大小和位置）。

使用大小属性设置 JFrame 或 JLabel 的尺寸

- 使用 setSize 设置以像素为单位的 JFrame 或 JLabel 中的宽度和高度。

设置内容面板的背景色

- 使用 setBackground 设置内容面板的背景属性。其中包括一些预定义的颜色：Color.RED, Color.GREEN 和 Color.BLUE 等。这些预定义的颜色完整列表见图 2.14。

设置 JLabel 的文本属性

- 使用 setText 设置 JLabel 的文本属性作为显示的信息。

设置 JLabel 的字体属性

- 使用 setFont 设置 JLabel 的字体属性，该属性可用来改变所显示文本的字体。可以指定字体的名称（如 SansSerif, Times, Courier 等）、字形（如 Font.PLAIN）以及字号大小。

对齐文本

- 使用 setHorizontalAlignment 设置 JLabel 的 horizontalAlignment 属性。horizontalAlignment 属性的一些可能的值包括 JLabel.LEFT（默认值），JLabel.CENTER 和 JLabel.RIGHT。

向 JFrame 中添加一幅图片

- 使用 JLabel 显示图片。
- 使用 setIcon 设置 JLabel 的图标属性显示图片。

关键术语

背景属性 用于指明内容面板或组件背景颜色的属性。

书名大写形式 文本中每个重要单词的首字母应作为大写的一种格式（如 Calculate the Total）。

范围属性 用于指明某组件位置和大小属性。

区分大小的 区别对待代码中的大写及小写字母。

.class 文件 由 Java 运行环境（JRE）执行的文件类型。class 文件是通过编译应用程序的.java 文件而产生的。

编译 将源代码文件（.java）转换成.class 文件的过程。

内容面板 用于包含 GUI 组件且属于 JFrame 的一个部分。

调试 查找并消除应用程序错误的过程。

dir 命令 通过在命令提示符窗口中输入以列出目录内容的一个命令。

字体属性 用于指明任何 JFrame 中所显示的文本或作为其内部组件进行显示的字体名称（如 SansSerif, Times, Courier 等）、字型（如 Font.PLAIN）和字号大小（如 12, 18, 36, 等等）的一种属性。

horizontalAlignment 属性 用于指明 JLabel 中文本对齐方式的一种属性。

图标属性 用于指明 JLabel 中所显示图片文件的一种属性。

javac 命令 将源代码文件 (.java) 编译成.class 文件的命令。

.java 文件 一种用于保存程序员编写的应用程序代码的文件类型。

JLabel 用于显示任何不允许用户进行修改的文本或图片的组件。

位置属性 指明出现在 JFrame 中某组件的左上角位置的属性。

逻辑错误 不能阻止应用程序成功编译, 但会使应用程序在运行时产生错误结果的一种错误。

像素 计算机屏幕上的一个点。像素是“图像元素”的简称。

RGB 值 用于创建某种颜色的红、绿、蓝数值。

分号(;) 用来表明 Java 语句结束的一种符号。

大小属性 以像素为单位指明某组件宽度和高度的属性。

源代码文件 具有.java 扩展名的一种文件。这些文件可由程序员编辑, 但不能够执行。

语句 指挥计算机完成某项任务的代码。每条语句是以一个分号(;) 作为结束。大多数应用程序是由多条语句组成的。

字符串 代表文本信息的字符序列。

字符串文字 位于双引号中的字符序列。

语法错误 当代码违反了某种编程语言的语法规则时所出现的一种错误。

文本属性 用于表示 JLabel 中所显示文本的一种属性。

标题栏 属于 JFrame 中用于显示标题的顶部区域。

GUI 设计导航

总体设计

可以在应用程序中使用不同的颜色, 但不应超出分散用户注意力的程度。

JFrame

- 选择简短的具备描述性质的 JFrame 标题。
- JFrame 的标题应使用书名大写形式, 即每个重要单词的首字母应为大写, 且不应使用任何标点符号作为结束。

JLabel

- 使用 JLabel 显示不允许用户修改的文本。
- 保证所有的 JLabel 组件能够足以显示它们的文本。
- 利用带有图片的 JLabel 可以改善需要使用一些不允许用户改动图形的 GUI。
- 保证所有的 JLabel 组件能够足以显示它们的图片。

Java 类库索引

JFrame 该组件将使一个 Java 应用程序出现在窗口中。应用程序中其他所有的组件均会显示在应用程序的窗口中。

- 运行



- 方法

setBackground 设置 JFrame 内容面板（或者是其他组件）的背景颜色。例如，如果 JFrame 的内容面板为 `contentPane`，则语句 `contentPane.setBackground(Color.YELLOW)`；会将内容面板的背景色设置为黄色。

setSize 设置 JFrame 的大小（以像素为单位）。

setTitle 设置 JFrame 标题栏中的文本。

JLabel 该组件显示不允许用户修改的文本和图片。

● 运行

Welcome to Java Programming!



● 方法

setBounds 指定 JLabel 的大小和位置。

setFont 指定 JLabel 中所显示文本的字体名、字形和字号大小。

setHorizontalAlignment 决定 JLabel 中文本的对齐方式。

setIcon 指定 JLabel 中图片的文件名和路径。

setSize 指定 JLabel 的高度和宽度（以像素为单位）。

setText 指定 JLabel 中显示的文本。

习题

选择题

- 2.1 利用 _____ 可以指定内容面板的背景色。
a) setBackground b) setBackColor c) setRGB d) setColor
- 2.2 为了编译应用程序，输入命令 _____ 并跟上相应文件的名称。
a) build b) compile c) javac d) create
- 2.3 字体属性用于控制 JLabel 中所显示字体的 _____。
a) 大小 b) 字形 c) 名字 d) 以上所有答案
- 2.4 语法错误可由 _____ 发现。
a) JRE b) 编译程序 c) 命令提示符 d) 应用程序
- 2.5 Java 的源代码文件具有 _____ 的扩展名。
a) .class b) .java c) .javac d) .source
- 2.6 范围属性用来控制 _____。
a) 文本的大小 b) 组件的位置 c) 组件的大小 d) (b)和(c)
- 2.7 一个 JLabel 组件通过指定 _____ 显示文本。
a) setCaption b) setData c) setText d) setName
- 2.8 使用 _____ 可以对齐 JLabel 中的文本。
a) setAlignment b) setCenter c) setRight d) setHorizontalAlignment
- 2.9 RGB 值可用于指定 _____。
a) JLabel 的大小 b) 某个颜色组件 c) 窗口的大小 d) JFrame 中的组件
- 2.10 像素是 _____。
a) 图像上的一个元素 b) 某种字体的测量方法 c) 字体集 d) 决定某个组件位置的属性

练习题

在习题 2.11 至习题 2.16 中，读者将使用本教程里的一些编程技术自定义各种不同的 GUI。这里用做自定义的应用程序并不能响应用户的操作。读者将在以后的教程中学习如何使这些应用程序具备完整的操作。

- 2.11（计算器 GUI）许多组件和 JLabel 一样，具备相同种类的一些属性。例如，JButton 也具有大小、位置和文本属性。在这个习题中，读者将在计算器 GUI 中自定义一个加法（+）JButton，如图 2.35 所示。

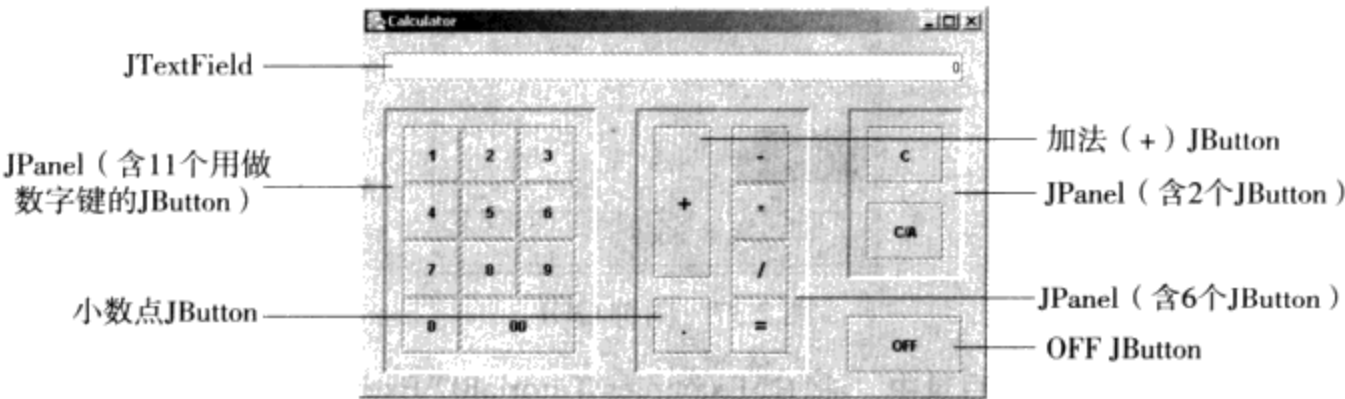


图 2.35 应用程序完成后的计算器 GUI

- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial02\Exercises\Calclater 目录复制到 C:\SimplyJava 目录中。
- b) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\Calculator 并按下 Enter 键进入到当前的工作目录中。
- c) 编译模板应用程序 通过键入命令 javac Calculator.java 并按下 Enter 键，对该应用程序进行编译。
- d) 运行模板应用程序 通过键入 java Calculator 运行这个应用程序。计算器模板应用程序的 GUI 将出现，如图 2.36 所示。注意其上缺少一个加法 (+) JButton。此模板应用程序虽然创建了加法 (+) JButton，但 JButton 默认的宽度和高度均为零，因此屏幕上无法将它显示出来。读者需要自定义这个加法 (+) JButton 的文本和范围属性并使它如图 2.35 那样显示出来。

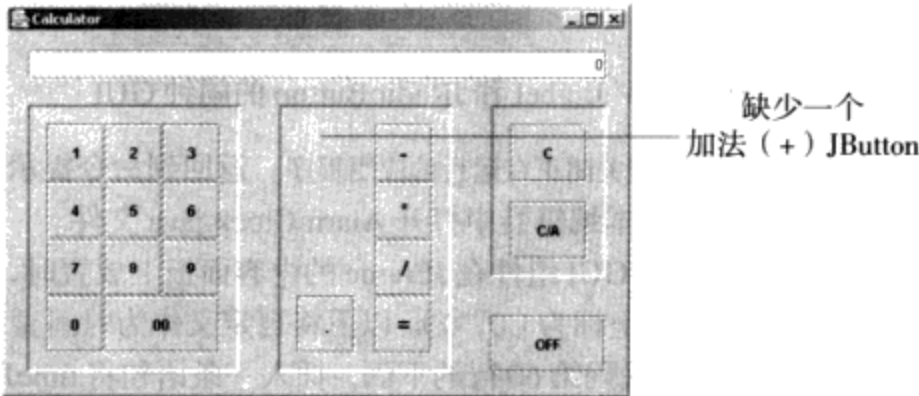


图 2.36 缺少加法 (+) JButton 的计算器 GUI

- e) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。返回到命令提示符窗口下。
 - f) 打开模板文件 在自己的文本编辑器中打开 Calculator.java 文件。
 - g) 在中间的 JPanel 中自定义加法 JButton 当某个 GUI 组件在 JPanel 上放置时，该组件将处在 JPanel 的左上角位置上，而此处坐标为 (0, 0)。为实现加法功能，需自定义这样的一个小 JButton。此 JButton 的名字为 plusJButton。在模板代码第 122 行下面，插入一条语句将 plusJButton 的文本属性设定为 "+"。接着下一行，插入一条语句将 plusJButton 的范围属性设定为 16, 16, 48, 128。
 - h) 保存应用程序 保存修改后的源代码文件。
 - i) 编译完成后的应用程序 在命令提示符窗口中，通过键入 javac Calculator.java 并按下 Enter 键对应用程序进行编译。
 - j) 运行完成后的应用程序 若应用程序能够正确编译，通过键入 java Calculator 运行它。将这个完成后的计算器应用程序的 GUI 同图 2.35 中所示的 GUI 做一比较，以确信正确地自定义了加法 (+) JButton。
 - k) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。返回到命令提示符窗口下。
 - l) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。
- 2.12 (闹钟 GUI) 在习题 2.11 中读者已经看到，JButton 有一些同 JLabel 一样的属性。在这个习题中，读者需要自定义作为 AM/PM 的 JRadioButton 组件的属性。除此之外，读者还将通过设置 JPanel 的背景属性为黑色以及前景属性为白色 (使用 setForeground) 作为时间显示的自定义。当读者完成修改以后，闹钟 GUI 会显示如图 2.37 中的样子。

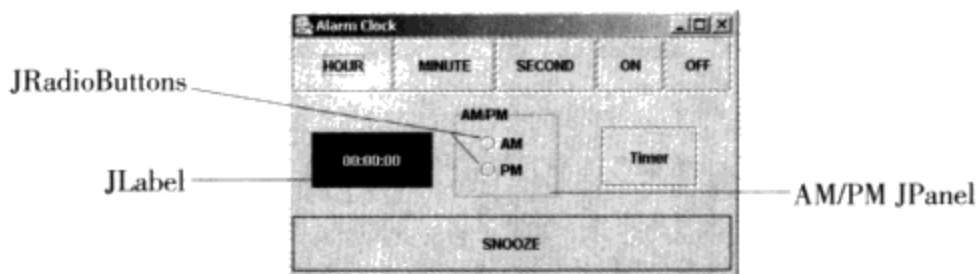


图 2.37 闹钟 GUI

- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial02\Exercises\AlarmClock 目录复制到 C:\SimplyJava 目录中。
- b) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。通过键入 `cd C:\SimplyJava\AlarmClock` 并按下 Enter 键进入到当前的工作目录中。
- c) 编译模板应用程序 通过键入命令 `javac AlarmClock.java` 并按下 Enter 键, 对该应用程序进行编译。
- d) 运行模板应用程序 通过键入 `java AlarmClock` 运行此应用程序。闹钟模板应用程序的 GUI 将会出现, 如图 2.38 所示。

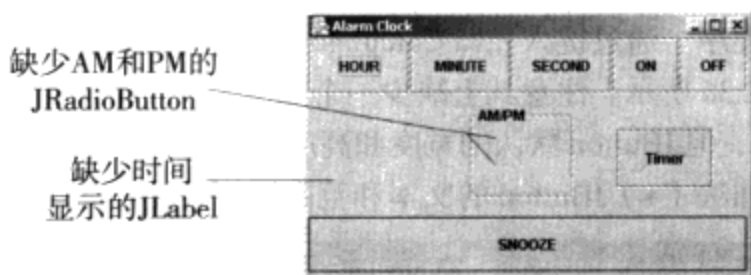


图 2.38 缺少 JLabel 和 JRadioButton 的闹钟 GUI

- e) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。返回到命令提示符窗口下。
- f) 打开模板文件 在自己的文本编辑器中打开 AlarmClock.java 文件。
- g) 自定义时间 JLabel 当某个 GUI 组件在 JFrame 的内容面板上放置时, 该组件将处在内容面板左上角的位置上, 且该处的坐标为 (0, 0)。以下将自定义作为时间显示的 JLabel。此 JLabel 的名字为 timeJLabel。在模板代码第 60 行的下面, 插入一条语句将 timeJLabel 的文本属性设置为 "00:00:00"。接着下一行, 插入一条语句将 timeJLabel 的范围属性设置为 16, 80, 100, 46。再下一行, 通过使用 setForeground 将 timeJLabel 的前景属性设置为 Color.WHITE。接着再下一行, 插入一条语句将 timeJLabel 的背景属性设置为 Color.BLACK (注意: 大多数的 GUI 组件都拥有前景和背景属性)。
- h) 在 AM/PM JPanel 中自定义 AM JRadioButton 在习题 2.11 中, 读者已经了解到当某个 GUI 组件在 JPanel 上放置时, 该组件将处在 JPanel 左上角的位置上, 且此处的坐标为 (0, 0)。为 AM 部分自定义 JRadioButton。将它命名为 amJRadioButton。在模板代码第 78 行后面, 插入一条语句将 amJRadioButton 的文本属性设置为 "AM"。接着下一行, 插入一条语句将 amJRadioButton 的范围属性设置为 20, 18, 50, 30。
- i) 在 AM/PM JPanel 中自定义 PM JRadioButton 为 PM 部分自定义 JRadioButton。将它命名为 pmJRadioButton。在模板代码第 84 行后面, 插入一条语句将 pmJRadioButton 的文本属性设置为 "PM"。接着下一行, 插入一条语句将 pmJRadioButton 的范围属性设置为 20, 40, 50, 30。
- j) 保存应用程序 保存修改后的源代码文件。
- k) 编译完成后的应用程序 在命令提示符窗口中, 通过键入 `javac AlarmClock.java` 并按下 Enter 键对应用程序进行编译。
- l) 运行完成后的应用程序 若应用程序能正确编译, 通过键入 `java AlarmClock` 运行它。将完成的闹钟应用程序的 GUI 同图 2.37 中所示的 GUI 做一比较, 以确信正确地自定义了 JLabel 和 JRadioButton。
- m) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。返回到命令提示符窗口下。

n) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

2.13 (微波炉 GUI) 同其他 GUI 组件一样, JPanel 也拥有范围和背景属性。在这个习题中, 读者将自定义一个代表微波炉门的 JPanel 的范围和背景属性。完成后的 GUI 如图 2.39 所示。

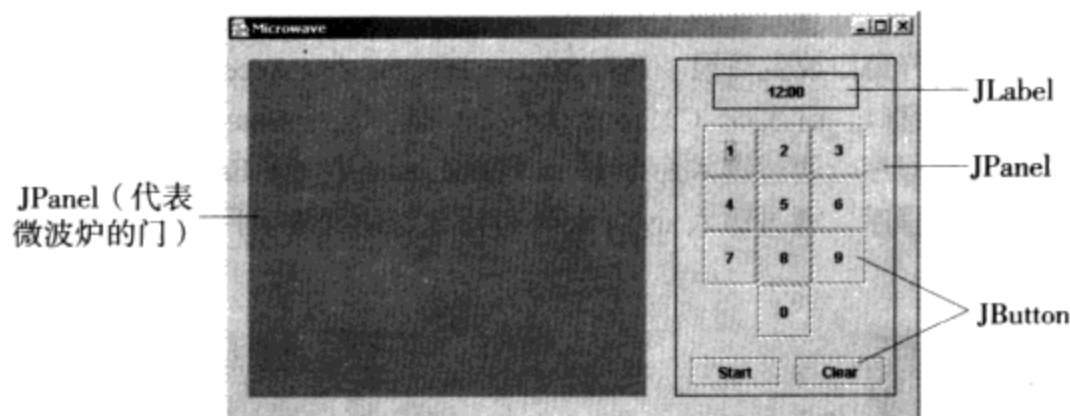


图 2.39 微波炉 GUI

- 将模板复制到工作目录中 将 C:\Examples\Tutorial02\Exercises\MicrowaveOven 目录复制到 C:\SimplyJava 目录中。
- 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\MicrowaveOven` 并按下 Enter 键进入到当前的工作目录中。
- 编译模板应用程序 通过键入命令 `javac Microwave.java` 并按下 Enter 键, 对该应用程序进行编译。
- 运行模板应用程序 通过键入 `java MicrowaveOven` 运行此应用程序。微波炉模板应用程序的 GUI 将会出现, 如图 2.40 所示。



图 2.40 缺少代表门 JPanel 的微波炉 GUI

- 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。返回到命令提示符窗口下。
- 打开模板文件 在自己的文本编辑器中打开 `Microwave.java` 文件。
- 在内容面板中自定义门 JPanel 自定义的这个 JPanel 用做微波炉门的显示。这个 JPanel 的名字是 `doorJPanel`。在第 31 行的后面插入一条语句将 `doorJPanel` 的范围属性设置为 16, 16, 328, 284。接着下一行, 插入一条语句将 `doorJPanel` 的背景属性设置为 `Color.GRAY`。
- 保存应用程序 保存修改后的源代码文件。
- 编译完成后的应用程序 在命令提示符窗口中, 通过键入 `javac Microwave.java` 并按下 Enter 键对应用程序进行编译。
- 运行完成后的应用程序 若应用程序能正确编译, 通过键入 `java Microwave` 运行它。将完成的微波炉应用程序的 GUI 同图 2.39 中所示的 GUI 做一比较, 以确信正确地自定义了 JPanel。
- (选做) 将门 JPanel 的背景色改为黄色 接下来, 读者将修改微波炉应用程序, 模拟出一个真实的微波炉。即当开始烹调时, 微波炉中的灯光会打开。需要读者将微波炉的背景色改成黄色, 模拟一个正在工作的微波炉。
- 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。返回到命令提示符窗口下。
- 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

2.14 (手机GUI)在这个习题中,读者将对手机数字键及其小键盘上的按钮位置做一调整,以显示如图 2.41 所示的结果。

- 将模板复制到工作目录中 将 C:\Examples\Tutorial02\Exercises\Phone 目录复制到 C:\SimplyJava 目录中。
- 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\Phone` 并按下 Enter 键进入到当前的工作目录中。
- 编译模板应用程序 通过键入命令 `javac Phone.java` 并按下 Enter 键,对该应用程序进行编译。
- 运行模板应用程序 通过键入 `java Phone` 运行此应用程序。手机模板应用程序的GUI将会出现,如图 2.42 所示。

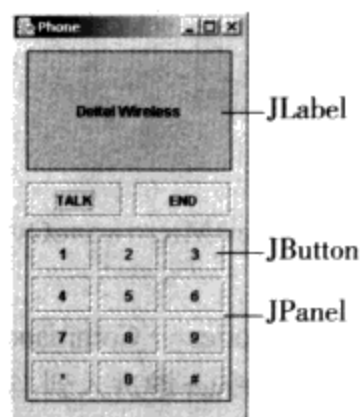


图 2.41 手机 GUI

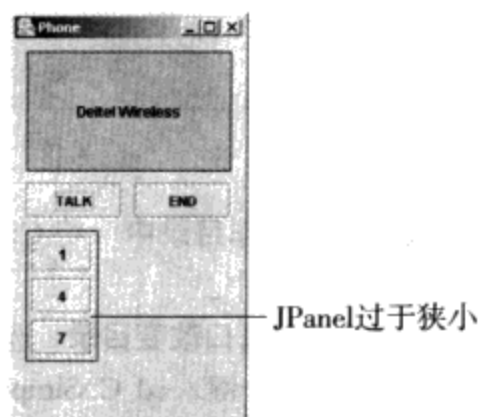


图 2.42 不具备正确小键盘格式的手机 GUI

- 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。返回到命令提示符窗口下。
- 打开模板文件 在自己的文本编辑器中打开 Phone.java 文件。
- 自定义小键盘 JPanel 很显然,这个小键盘 JPanel 对于小键盘上的所有按钮来说是过于狭小了。此 JPanel 的名字为 numberJPanel。在第 54 行,修改语句将 numberJPanel 的宽度和高度分别设置为 170 和 145。
- 自定义小键盘 JButton 在第 59 行至第 129 行中包含了一些用于指定小键盘上 JButton 属性的语句。找到这些设置每一个 JButton 边界属性的语句,并修改这一语句使之正确放置相应的 JButton。左侧一列的每一个 JButton 的 x 坐标应为 5。中间一列的每一个 JButton 的 x 坐标应为 60。而右侧一列的每一个 JButton 的 x 坐标则为 115。第 1 行的每一个 JButton 的 y 坐标应为 5。第 2 行上的每一个 JButton 的 y 坐标应为 40。第 3 行上的每一个 JButton 的 y 坐标为 75。而第 4 行上的每一个 JButton 的 y 坐标则为 110。
- 保存应用程序 保存修改后的源代码文件。
- 编译完成后的应用程序 在命令提示符窗口中,通过键入 `javac Phone.java` 并按下 Enter 键对应用程序进行编译。
- 运行完成后的应用程序 若应用程序能正确编译,通过键入 `java Phone` 运行它。将完成后的手机应用程序的 GUI 同图 2.41 中所示的 GUI 做一比较,以确信正确地自定义了 JPanel 和 JButton。
- 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。返回到命令提示符窗口下。
- 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

2.15 (自动售货机GUI)在这个习题中,读者将自定义一些用于显示图片的JLabel的属性(如图2.43所示)。

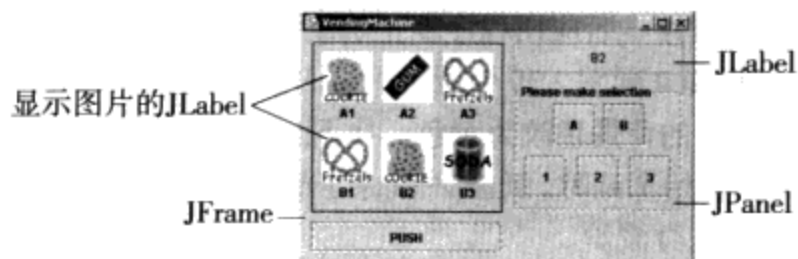


图 2.43 自动售货机 GUI

- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial02\Exercises\VendingMachine 目录复制到 C:\SimplyJava 目录中。
- b) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\VendingMachine 并按下 Enter 键进入到当前的工作目录中。
- c) 编译模板应用程序 通过键入命令 javac VendingMachine.java 并按下 Enter 键，对该应用程序进行编译。
- d) 运行模板应用程序 通过键入 java VendingMachine 运行此应用程序。自动售货机模板应用程序的 GUI 将会出现，如图 2.44 所示。

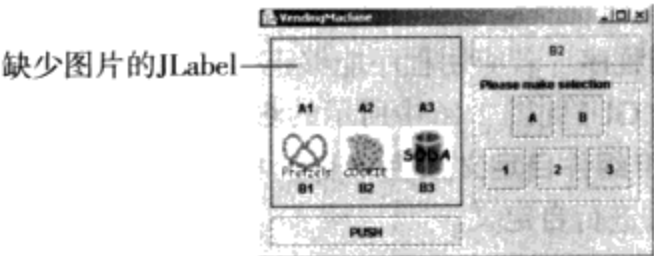


图 2.44 缺少三幅图片的自动售货机 GUI

- e) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。返回到命令提示符窗口下。
- f) 打开模板文件 在自己的文本编辑器中打开 VendingMachine.java 文件。
- g) 自定义一个 JLabel 的图标 读者需要给位于 A1 之上的 JLabel 指定一个图标。此 JLabel 的名字为 a1IconJLabel。在第 39 行的下面，插入一条语句将 a1IconJLabel 的图标属性设置为 "images/cookie.png"（提示：可以使用类似图 2.31 中第 36 行里的语句）。
- h) 自定义位于 A2 和 A3 之上 JLabel 的图标 读者还需要为 GUI 中位于 A2 和 A3 之上的 JLabel 指定图标。这些 JLabel 的名字分别是 a2IconJLabel 和 a3IconJLabel。在第 52 行的下面，插入一条语句将 a2IconJLabel 的图标属性设置为 "images/gum.png"。在第 65 行的下面，插入一条语句将 a3IconJLabel 的图标属性设置为 "images/pretzel.png"。
- i) 保存应用程序 保存修改后的源代码文件。
- j) 编译完成后的应用程序 在命令提示符窗口中，通过键入 javac VendingMachine.java 并按下 Enter 键对应用程序进行编译。
- k) 运行完成后的应用程序 若应用程序能正确编译，通过键入 java VendingMachine 运行它。将完成后的自动售货机应用程序的 GUI 同图 2.43 中所示的 GUI 做一比较，以确信正确地自定义了 JLabel。
- l) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。返回到命令提示符窗口下。
- m) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

挑战题

2.16（收音机 GUI）GUI 设计的一个重要部分是对颜色的选择。图 2.45 中显示的这个收音机 GUI 使用了一些不太吸引人的颜色。从图 2.14 中所列的表中选取一些预先定义好的 Color 值，对这个收音机 GUI 中的颜色进行修改，使之看起来更加吸引人。比如说，图 2.46 中显示的这个收音机 GUI 是将所有的桔黄色替换成了青色，并将所有的粉红色替换成了浅灰色。读者可以尝试更多颜色的选择。

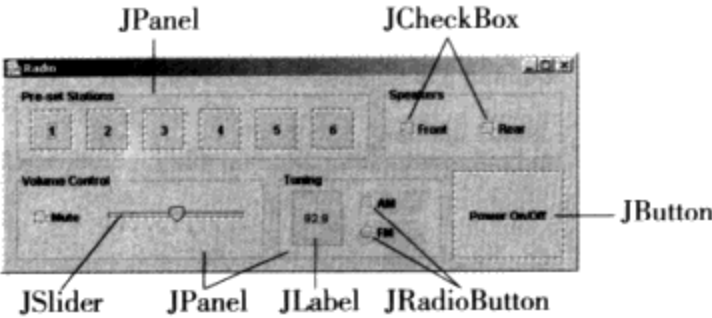


图 2.45 收音机 GUI

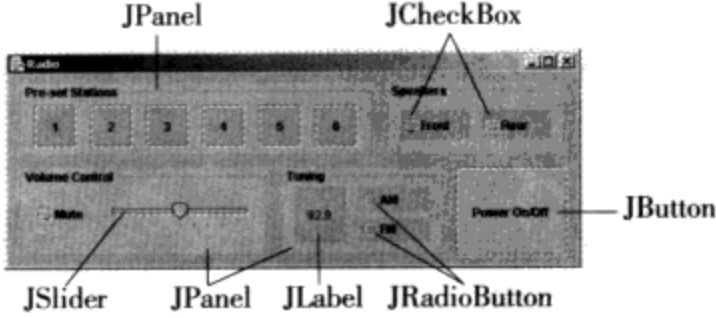


图 2.46 拥有不同颜色的收音机 GUI

- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial02\Exercises\Radio 目录复制到 C:\SimplyJava 目录中。
- b) 打开模板文件 在自己的文本编辑器中打开 Radio.java 文件。
- c) 自定义颜色 在模板代码中, 凡出现 Color.ORANGE 的地方均替换成 Color.CYAN。然后, 将凡出现 Color.PINK 的地方替换成 Color.LIGHT_GRAY。
- d) 保存应用程序 保存修改后的源代码文件。
- e) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\Radio` 并按下 Enter 键进入到当前的工作目录中。
- f) 编译完成后的应用程序 在命令提示符窗口中, 通过键入 `javac Radio.java` 并按下 Enter 键对该应用程序进行编译。
- g) 运行完成后的应用程序 若应用程序能够正确编译, 通过输入 `java Radio` 运行它。将完成后的收音机应用程序的 GUI 同图 2.46 中所示的 GUI 做一比较, 以确信正确地自定义了颜色。
- h) (选做) 为每个 GUI 组件自定义颜色 读者可按照自己的喜好从图 2.14 中选择一些不同的颜色并对这些 GUI 组件进行自定义。



教程3 库存清单应用程序

介绍 JTextField 和 JButton 组件

教学目标

在本教程中，读者将学到以下内容：

- 使用图形用户界面设计原则创建 GUI
- 在应用程序窗口中自定义 JLabel, JTextField 和 JButton
- 水平对齐 JTextField 中的文本
- 将某个 JTextField 指定为不可编辑

本教程将向读者介绍一些 GUI 组件并开始讨论 GUI 设计方面的一些相关问题。读者将设计一个简单库存清单应用程序的 GUI，通过实现每一步及自定义相应组件，对这个应用程序的用户界面进行完善。还将学到 JLabel 的一些新属性，同时自定义几个 JTextField 和 JButton。最后，这些组件都会出现在一个 JFrame 中。在本教程的结尾处，将会看到一系列有关 GUI 设计的指导原则，利用这一指导原则为读者将来创建出吸引人且易于使用的图形用户界面提供帮助。我们还将附录 C 中对本书内的所有 GUI 设计原则进行总结。

3.1 探试库存清单应用程序

在本教程中，读者将设计一个库存清单应用程序的图形用户界面，该程序用于计算某大学书店所收教材的总数量。这个应用程序必须满足下面的需求：

应用程序需求分析

某大学书店收到几箱教材。在一批教材中，每箱教材的数量是相同的。仓库管理员希望使用一台计算机来计算书店收到的每批教材的总数量。仓库管理员会输入一批教材中箱子的数目及每只箱子中所存放教材的数目；然后该应用程序将计算并显示出这批教材的总数量。

此应用程序将执行一个简单的计算。用户（仓库管理员）通过 JTextField 输入箱子的数目以及箱中所存放教材的数目。然后点击一个 JButton，这时应用程序将两个数相乘并在 JTextField 中显示出所收到教材的总数量。在本教程中，读者将在一个 JFrame 中自定义一个 JTextField 和一个 JButton。在教程 4 中，还将通过添加用户点击 JButton 时的执行代码，对这一库存清单应用程序进行完善。读者先以这个完成后的应用程序的探试作为起点。然后，学习一些 Java 技术，创建一个属于自己的应用程序。



探试完成后的库存清单应用程序

1. 定位到完成后的应用程序 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\Examples\Tutorial03\CompletedApplication\Inventory` 然后按下 Enter 键，将目录改变到这个完成后的库存清单应用程序的目录下面。

- 2. 打开库存清单应用程序的模板文件 在文本编辑器中打开模板文件 Inventory.java。
 - 3. 设置 JLabel 的文本、大小和位置 将图 3.5 中的第 40 行和第 41 行插入到应用程序中。第 40 行使用 setText 将 cartonsJLabel 的文本属性设置为 "Cartons per shipment:"。当指定某个 JLabel 的文本属性值时，应使用语句大写形式，即对文本中的第一个单词以及每一个有意义的名词的第一个字母采用大写的形式（如 Sales for January:）。
- 第 41 行是设置 cartonsJLabel 的范围属性，以便在 JFrame 的内容面板中放置这样的 一个 JLabel。回顾教程 2 中曾使用过 setBounds 设置某个组件的位置和大小。这个 JLabel 的位置为 16, 16，其宽度和高度分别为 130, 21。读者也可以使用 setLocation 和 setSize 分别设置某个组件的位置和大小。

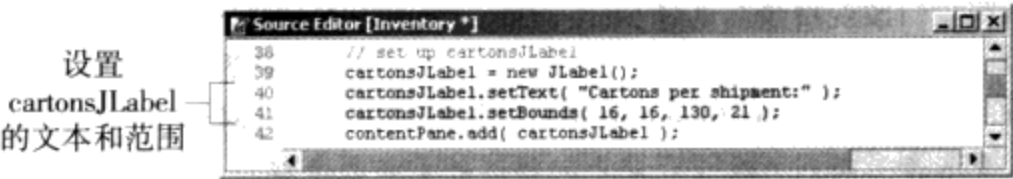


图 3.5 设置 Cartons per shipment: JLabel 的文本和范围



GUI 设计提示

起描述性作用的 JLabel 应使用语句大写形式并以冒号结束。



GUI 设计提示

将每一个起描述性作用的 JLabel 放置在所标识组件（如 JTextField）的上方或者是左侧。

- 4. 保存应用程序 保存修改后的源代码文件。
 - 5. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\Inventory 并按下 Enter 键进入到工作目录中。
 - 6. 编译应用程序 键入命令 javac Inventory.java 并按下 Enter 键，对该应用程序进行编译。如果没有编译错误，进入第 7 步。否则，改正代码中的错误并重复本步骤。
 - 7. 运行应用程序 若此应用程序能正确编译，可以通过键入 java Inventory 并按下 Enter 键来运行它。
- 图 3.6 显示的是更新后的应用程序的运行结果。

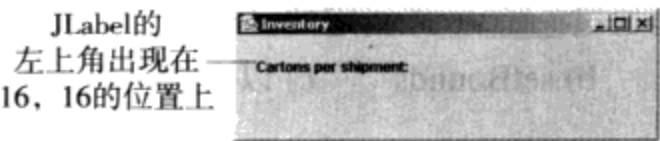


图 3.6 更新了 Cartons per shipment:JLabel 的应用程序

- 8. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- 9. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

接下来，读者将自定义余下的 JLabel，这个过程将帮助读者理解：怎样才可以提供输入及怎样去解释应用程序的输出结果。本教程的后面部分，读者还将自定义这些 JLabel 所标识的组件。

自定义其余的 JLabel

- 1. 自定义第二个具备描述性质的 JLabel 插入图 3.7 中的第 46 行和第 47 行。第 46 行是将 itemsJLabel 的文本设置为 "Items per carton:"。第 47 行是将 itemsJLabel 的范围设置为 16, 48, 104, 21。注意两个 JLabel（Items per carton:和 Cartons per shipment:）距离窗口左边界具有相同的距离（16 像素）。这使得这些 JLabel 能够垂直对齐。
- 2. 自定义第三个具备描述性质的 JLabel 插入图 3.8 中的第 52 行和第 53 行。第 52 行是将 totalJLabel 的文本设置为 "Total:"。第 53 行是将 totalJLabel 的范围设置为 204, 16, 40, 21。

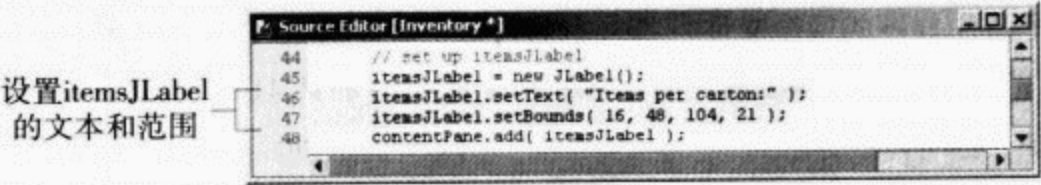


图 3.7 设置 Items per carton:JLabel 的文本和范围



GUI 设计提示

如果 JLabel 是垂直放置的，应使这一组起描述性作用的 JLabel 的左边界对齐。

- 3. 保存应用程序 保存修改后的源代码文件。
 - 4. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\Inventory 并按下 Enter 键进入到工作目录中。
 - 5. 编译应用程序 键入命令 javac Inventory.java 并按下 Enter 键，对该应用程序进行编译。如果没有编译错误，进入第 6 步。否则，改正代码中的错误并重复本步骤。
 - 6. 运行应用程序 若此应用程序能正确编译，可以通过键入 java Inventory 并按下 Enter 键来运行它。
- 图 3.9 显示的是更新后的应用程序的运行结果。读者还将添加其他一些缺少的组件。

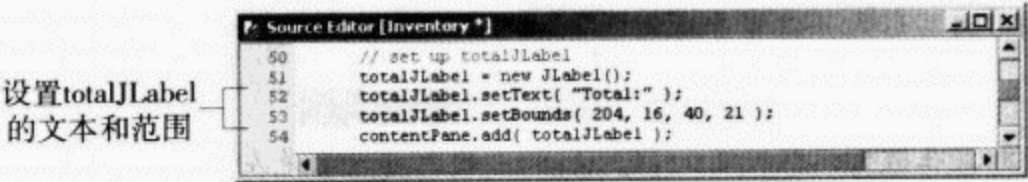


图 3.8 设置 Total:JLabel 的文本和范围

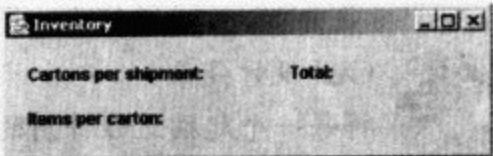


图 3.9 运行已自定义了 3 个 JLabel 的库存清单应用程序

- 7. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- 8. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

自测题

- 1. JLabel 上的文本是通过 _____ 来指定的。
a) setLabel b) changeLabel c) setText d) changeText
- 2. JLabel 的范围和大小是通过 _____ 来指定的。
a) setSizeAndLocation b) setBounds c) 以上选项均不对 d) a 和 b

答案：1) c 2) b

3.3 自定义库存清单应用程序的 JTextField 和 JButton



GUI 设计提示

利用 JTextField 接收键盘中输入的数据。

库存清单应用程序利用用户输入的数据计算一批货物中教材的总数量。具体地说，用户需要输入箱子的数目及每箱所存放的教材数量。利用 JTextField 组件接收键盘中输入的这些数据。接下来，读者将学习如何去自定义 JTextField 组件。

自定义 JTextField 组件

- 1. 自定义 JTextField 设置 JTextField 的属性与设置 JLabel 的属性类似。回到文本编辑器中，将图 3.10 中的第 58 行至第 60 行插入到代码中。第 58 行使用 setText 将 Cartons per shipment:JTextField 的文本属性设置为 "0"。这将作为应用程序在运行时一个默认的显示值。第 59 行使用 setBounds 将范围属性设置

为 148, 16, 40, 21。利用这些位置和大小值将 JTextField 的顶部与描述它的 Cartons per shipment:JLabel 的顶部对齐（值 16 表示的是 JTextField 的垂直位置）。第 60 行使用 setHorizontalAlignment 将 JTextField 的 horizontalAlignment 属性设置为 JTextField.RIGHT，这会使 JTextField 的文本靠右对齐。文本也可以靠左对齐（JTextField.LEFT）或居中（JTextField.CENTER）。



GUI 设计提示

每一个 JTextField 都应该有一个指明其意图的起描述性作用的 JLabel。

设置cartonsJTextField的
文本、范围和对齐方式

```
Source Editor [Inventory *]
56 // set up cartonsJTextField
57 cartonsJTextField = new JTextField();
58 cartonsJTextField.setText( "0" );
59 cartonsJTextField.setBounds( 148, 16, 40, 21 );
60 cartonsJTextField.setHorizontalAlignment( JTextField.RIGHT );
61 contentPane.add( cartonsJTextField );
```

图 3.10 设置 Cartons per shipment:JTextField 的属性



GUI 设计提示

如果可能的话，为所期望的输入提供足够宽的 JTextField。否则，文本会在用户输入时发生滚动并只显示出其中的一部分。

- 2. 自定义第二个 JTextField 将图 3.11 中的第 65 行至第 67 行插入到代码中。第 65 行是将 Items per carton: JTextField 的文本设置为 "0"，用其作为应用程序开始执行时的 JTextField 中的显示。第 66 行是将 JTextField 的范围设置为 148, 48, 40, 21。这些值用来保证 Cartons per shipment:和 Items per carton:这两个 JTextField 的左边界能够对齐。这些值也同样用于保证 Items per carton:JLabel 的顶部与 Items per carton:JTextField 的顶部对齐。第 67 行是将 JTextField 中的文本向右对齐。



GUI 设计提示

对齐垂直排列的 JTextField 的左边界。

设置itemsJTextField的
文本、范围和对齐方式

```
Source Editor [Inventory *]
63 // set up itemsJTextField
64 itemsJTextField = new JTextField();
65 itemsJTextField.setText( "0" );
66 itemsJTextField.setBounds( 148, 48, 40, 21 );
67 itemsJTextField.setHorizontalAlignment( JTextField.RIGHT );
68 contentPane.add( itemsJTextField );
```

图 3.11 设置 Items per carton:JTextField 的属性



GUI 设计提示

一般来说，JTextField 中的数字应该靠右对齐。

- 3. 自定义输出 JTextField 将图 3.12 中的第 72 行至第 74 行插入到应用程序中。第 72 行是将 JTextField 的范围设置为 244, 16, 86, 21。第 73 行和第 74 行中的语句是将 JTextField 中的文本靠右对齐。这里并没有为作为输出的 JTextField 设置任何默认文本（即 JTextField 在显示时所出现的文本），因而最初的 JTextField 便是空白的。

设置totalResultJTextField
的范围和对齐方式

```
Source Editor [Inventory *]
70 // set up totalResultJTextField
71 totalResultJTextField = new JTextField();
72 totalResultJTextField.setBounds( 244, 16, 86, 21 );
73 totalResultJTextField.setHorizontalAlignment(
74     JTextField.RIGHT );
75 contentPane.add( totalResultJTextField );
```

图 3.12 设置 Total:JTextField 的范围及水平对齐方式

- 4. 使输出 JTextField 为不可编辑的 将图 3.11 中的第 75 行插入到应用程序中。此行使用 setEditable 将 Total:JTextField 的可编辑属性设置为 false。一旦可编辑属性为 false，则 JTextField 将变成灰色，表明用户在这个 JTextField 中是不能对任何现有的文本进行编辑或是输入任何新的文本。使用不可编辑的

JTextField也可用来向用户显示信息。尽管用户不能改变不可编辑的JTextField中的文本,但应用程序却可以改变JTextField中所显示的内容。而对于那些在显示以后便不再被改动的文本来说,使用JLabel则比较合适。



GUI 设计提示

将应用程序输出组件置于应用程序输入组件的下方和 / 或右侧。

将totalResultJTextField的可编辑属性设置为false防止用户对它进行修改

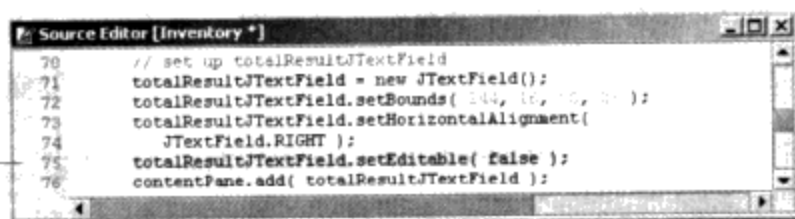


图 3.13 改变 TotalJTextField 的可编辑属性

5. 保存应用程序 保存修改后的源代码文件。
6. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\Inventory` 并按下 Enter 键进入到工作目录中。
7. 编译应用程序 键入命令 `javac Inventory.java` 并按下 Enter 键, 对该应用程序进行编译。如果没有编译错误, 进入第 8 步。否则, 改正代码中的错误并重复本步骤。



GUI 设计提示

作为输出的JTextField应该与作为输入的JTextField有所区别。将输出JTextField的可编辑属性设置为false可防止用户向JTextField中输入数据, 这同时也会使JTextField以灰色的背景出现在用户图形界面中。

8. 运行应用程序 若此应用程序能够正确编译, 可以通过键入 `java Inventory` 并按下 Enter 键来运行它。图 3.14 显示了更新后的应用程序的运行结果。

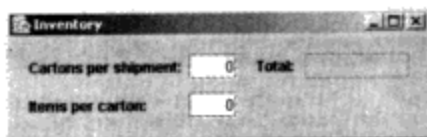


图 3.14 运行自定义了 JTextField 的库存清单应用程序

9. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
10. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。



GUI 设计提示

如果组件是按照水平方式进行放置的, 那么作为描述性质使用的JLabel应该与它所描述的组件具有相同的高度 (参见图 3.14)。

在图 3.14 中, 读者可能会注意到组件是按照水平和垂直方式对齐的。一般来说, 应该将每一个起描述性作用的JLabel置于所描述组件 (例如 JTextField) 的上方或者是它的左侧。如果组件是按照水平方式排列的 (即处于同一行), 则这个起描述性作用的JLabel应和它所描述的组件拥有相同的高度。如果组件是按照垂直方式排列的 (即一个在另一个的上方), 则JLabel应放置在所描述组件的上方并且这些组件的左边界应该对齐。同样, JFrame中的每一组组件, 它们之间也应留有间隙。只要遵照这些简单的规则, 所创建出来的应用程序在外观上会变得更加吸引人, 而且也更容易使用。



GUI 设计提示

一个起描述性作用的JLabel和它所描述的组件, 如果它们是按照水平方式放置的, 则它们的顶部应该对齐。

现在读者已经自定义了应用程序中所有的JTextField。其中两个是作为输入JTextField，使用它们从用户那里收集数据。而第三个是作为输出JTextField，可用做向用户显示信息。



GUI 设计提示

一个起描述性作用的JLabel和它所描述的组件，如果它们是按照垂直方式放置的，则它们均应靠左对齐。

既然用户可以利用JTextField输入数据，还应该为用户提供一种用于指挥应用程序执行乘法计算的方式。一种最常用的方法是给用户提供一个用来点击的JButton。下面，读者将在库存清单应用程序中自定义这样的一个JButton。

自定义一个 JButton

- 1. 自定义一个 JButton 将图 3.15 中的第 80 行和第 81 行插入到代码中。第 80 行是利用 setText 将 JButton 的文本属性设置为 "Calculate Total"。JButton 的文本属性会将该值显示在 JButton 的表面上。读者应该为 JButton 的文本属性使用书名大写形式，即文本中每个重要单词的首字母应为大写（就像 Calculate Total 一样）。当标注 JButton 时，在保证文本尽可能简短的同时还应该清楚地表明 JButton 的功能。第 81 行使用 setBounds 将 JButton 的范围属性设置为 204, 48, 126, 24。这些值会使该 JButton 左右两边与上面的两个 Total:JLabel 和 Total:JTextField 对齐。



GUI 设计提示

JButton 利用文本属性进行标注。JButton 的文本应使用书名大写形式并且在力求简短的同时还能够向用户表达一定的含义。

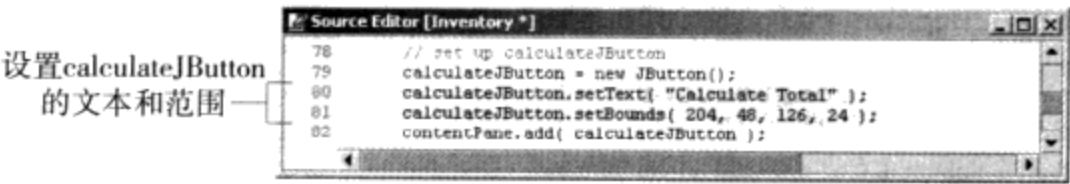


图 3.15 自定义 CalculateTotal JButton

- 2. 保存应用程序 保存修改后的源代码文件。



GUI 设计提示

多个JButton应从JFrame的顶部位置垂直向下排列，或者是在JFrame底部位置的某一行上水平地进行排列。

- 3. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\Inventory 并按下 Enter 键进入到工作目录中。
- 4. 编译应用程序 键入命令 javac Inventory.java 并按下 Enter 键，对该应用程序进行编译。如果没有编译错误，进入第 5 步。否则，改正代码中的错误并重复本步骤。
- 5. 运行应用程序 若此应用程序能正确编译，可以通过键入 java Inventory 并按下 Enter 键来运行它。图 3.14 显示了更新后的应用程序的运行结果。注意，此时如果点击 Calculate Total JButton，则不会起任何作用。这是因为读者还未编写用来告诉应用程序如何响应点击这个 JButton 的代码。在教程 4 中，读者将通过编写用户点击 JButton 继而会在 Total:JTextField 中显示一批货物中的图书总数量的代码来完成这个完整的库存清单应用程序。

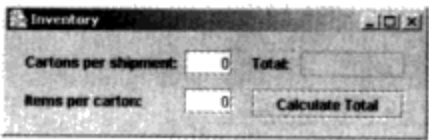


图 3.16 运行自定义了 Calculate Total JButton 的应用程序

6. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。

7. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

图 3.17 中给出了库存清单应用程序的完整源代码。本教程中，凡需要添加、查看或是修改的代码均在图中相应的代码行中做了突出显示。

```

1 // Tutorial 3: Inventory.java
2 // Calculates the number of items in a shipment based on the number
3 // of cartons received and the number of items per carton.
4 import java.awt.*;
5 import java.awt.event.*;
6 import javax.swing.*;
7
8 public class Inventory extends JFrame
9 {
10     // JLabel and JTextField for cartons per shipment
11     private JLabel cartonsJLabel;
12     private JTextField cartonsJTextField;
13
14     // JLabel and JTextField for items per carton
15     private JLabel itemsJLabel;
16     private JTextField itemsJTextField;
17
18     // JLabel and JTextField for total items per shipment
19     private JLabel totalJLabel;
20     private JTextField totalResultJTextField;
21
22     // JButton to initiate calculation of total items per shipment
23     private JButton calculateJButton;
24
25     // no-argument constructor
26     public Inventory()
27     {
28         createUserInterface();
29     }
30
31     // create and position GUI components; register event handlers
32     public void createUserInterface()
33     {
34         // get content pane and set layout to null
35         Container contentPane = getContentPane();
36         contentPane.setLayout( null );
37
38         // set up cartonsJLabel
39         cartonsJLabel = new JLabel();
40         cartonsJLabel.setText( "Cartons per shipment:" );
41         cartonsJLabel.setBounds( 16 , 16 , 130 , 21 );
42         contentPane.add( cartonsJLabel );
43
44         // set up itemsJLabel
45         itemsJLabel = new JLabel();
46         itemsJLabel.setText( "Items per carton:" );
47         itemsJLabel.setBounds( 16 , 48 , 104 , 21 );
48         contentPane.add( itemsJLabel );
49
50         // set up totalJLabel
51         totalJLabel = new JLabel();
52         totalJLabel.setText( "Total:" );
53         totalJLabel.setBounds( 204 , 16 , 40 , 21 );
54         contentPane.add( totalJLabel );
55
56         // set up cartonsJTextField

```

```

57     cartonsJTextField = new JTextField();
58     cartonsJTextField.setText( "0" );                                设置cartonsJTextField
59     cartonsJTextField.setBounds( 148, 16 , 40 , 21 );                的文本、范围
60     cartonsJTextField.setHorizontalAlignment( JTextField.RIGHT );      和对齐方式
61     contentPane.add( cartonsJTextField );
62
63     // set up itemsJTextField
64     itemsJTextField = new JTextField();
65     itemsJTextField.setText( "0" );                                设置itemsJTextField
66     itemsJTextField.setBounds( 148, 48, 40 , 21 );                的文本、范围
67     itemsJTextField.setHorizontalAlignment( JTextField.RIGHT );      和对齐方式
68     contentPane.add( itemsJTextField );
69
70     // set up totalResultJTextField
71     totalResultJTextField = new JTextField();
72     totalResultJTextField.setBounds( 244, 16 , 86 , 21 );          设置 totalResultJTextField
73     totalResultJTextField.setHorizontalAlignment(                    的范围、对齐方式
74         JTextField.RIGHT );                                         和可编辑性
75     totalResultJTextField.setEditable( false );
76     contentPane.add( totalResultJTextField );
77
78     // set up calculateJButton
79     calculateJButton = new JButton();
80     calculateJButton.setText( "Calculate Total" );                  设置 calculateJButton
81     calculateJButton.setBounds( 204 , 48, 126, 24 );              的文本和范围
82     contentPane.add( calculateJButton );
83
84     // set properties of application's window
85     setTitle( "Inventory" ); // set title bar text
86     setSize( 354, 112 ); // set window size
87     setVisible( true ); // display window
88
89 } // end method createUserInterface
90
91 // main method
92 public static void main( String[] args )
93 {
94     Inventory application = new Inventory();
95     application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
96
97 } // end method main
98
99 } // end class Inventory

```

图 3.17 库存清单应用程序的完整代码

自测题

1. 使用 _____ 设置 JButton 上的文本。

a) setName

b) setText

c) setTitle

d) setFace

2. JButton 应该使用 _____ 大写形式。

a) 书名

b) 语句

c) 按钮

d) 以上答案均不对

答案: 1) b 2) a

3.4 小结

在本教程中,读者通过设计图形用户界面创建了一个属于自己的库存清单应用程序。学习了如何使用 JLabel 来描绘组件。然后,通过自定义 JTextField,允许用户从键盘中输入数据并将输出结果再显示给用户。读者了解到若将 JTextField 的可编辑属性设置为 false,便能够表明该 JTextField

只可作为输出来使用。最后，通过在库存清单应用程序中自定义一个 JButton，从而允许用户向应用程序发出执行某个操作的信号（将对两个来自 JTextField 的数据进行相乘运算并显示其结果）。在 JFrame 中自定义组件时，读者还学到了一些 GUI 的设计准则，这有助于读者创建出吸引人的且非常直观的图形用户界面。

在下一个教程中，将向读者讲授如何利用 Java 编写诸如用户点击 Calculate Total JButton 时程序如何执行的代码。即，当 JButton 被点击时，应用程序会接收到一个称之为事件的信号。读者需要学习如何编写通过执行相乘运算及显示运算结果来响应该事件的应用程序。

技术小结

自定义一个起描述性质的 JLabel

- 在自定义一个 JLabel 时，通过设置其文本属性（使用 setText）和范围属性（使用 setBounds）使该 JLabel 同另一组件对齐。

自定义一个 JTextField

- 在自定义一个 JTextField 时，通过设置其文本属性（使用 setText）和范围属性（使用 setBounds）使该 JTextField 同另一组件对齐，并且还可以设置它的 horizontalAlignment 属性（利用 setHorizontalAlignment，其选项有：JTextField.LEFT，JTextField.CENTER，JTextField.RIGHT）。

自定义一个输出 JTextField

- 在自定义一个输出 JTextField 时，将可编辑属性设置为 false（使用 setEditable）。

自定义一个 JButton

- 在自定义一个 JButton 时，通过设置其文本属性（使用 setText）和范围属性（使用 setBounds）使该 JButton 同另一组件对齐。

关键术语

可编辑属性 用于指明 JTextField 外观及操作的属性，通过它可区分开输入 JTextField（可编辑属性为 true）和输出 JTextField（可编辑属性为 false）。

horizontalAlignment 属性 用于指明 JTextField 中文本对齐方式的属性（JTextField.LEFT，JTextField.CENTER，JTextField.RIGHT）。

输入 JTextField 作为获取用户输入的一种 JTextField。输入 JTextField 的可编辑属性被设置为 true，这也是它的默认值。

JButton 组件 一种在点击后可以命令应用程序完成某项操作的组件。

JLabel 组件 一种用来描述其他组件的组件。利用它可帮助用户理解某个组件的意图。

JTextField 组件 一种可从键盘中接受用户的输入或者是将输出显示给用户的组件。

JTextField.CENTER 同 setHorizontalAlignment 配合使用以居中 JTextField 中的文本。

JTextField.LEFT 同 setHorizontalAlignment 配合使用以靠左对齐 JTextField 中的文本。

JTextField.RIGHT 同 setHorizontalAlignment 配合使用以靠右对齐 JTextField 中的文本。

输出 JTextField 用于显示计算结果的一种 JTextField。输出 JTextField 是利用 setEditable 将它的可编辑属性设置为 false。

语句大写形式 将文本中首个单词的第一个字母作为大写（如 Cartons per shipment）的一种格式；文本中除了还需将特定名词的第一个字母也作为大写外，其他字母应为小写。

不可编辑 JTextField 不允许用户向其中输入值或是修改现有文本的一种 JTextField。这种 JTextField 通常用于显示计算的结果。

GUI 设计导航

JButton

- JButton 利用文本属性进行标注。JButton 上的文本应使用书名大写形式并且在尽量简短的同时还能够向用户表达一定的含义。
- 多个 JButton 应从 JFrame 的顶部位置垂直向下排列, 或者是在 JFrame 底部位置的某一行上水平地进行排列。

JFrame

- 将应用程序的输出组件置于应用程序输入组件的下方和 / 或右侧。

JLabel

- 利用 JLabel 标识其他 GUI 组件。
- 起描述性作用的 JLabel 应使用语句大写形式并以冒号结束。
- 将每一个起描述性作用的 JLabel 放置在所标识组件 (如 JTextField) 的上方或者是左侧。
- 如果 JLabel 是垂直放置的, 应将这一组起描述性作用的 JLabel 的左边界对齐。
- 如果组件是按照水平方式进行放置, 则起描述性作用的 JLabel 应该与它所描述的组件拥有相同的高度。
- 一个起描述性作用的 JLabel 和它所描述的组件, 如果它们是以水平方式进行放置, 则它们的顶部应该对齐。
- 一个起描述性作用的 JLabel 和它所描述的组件, 如果它们是按照垂直的方式进行放置, 则它们均应靠左对齐。

JTextField

- 利用可编辑的 JTextField 从键盘中输入数据。默认情况下 JTextField 是可编辑的。
- 每一个 JTextField 应该有一个指明其意图的起描述性作用的 JLabel。
- 如果可能的话, 为所期望的输入提供足够宽的 JTextField。否则, 文本会在用户输入时发生滚动并只显示出其中的一部分。
- 对齐垂直排列的 JTextField 的左边界。
- 一般来说, JTextField 中的数字应该靠右对齐。
- 作为输出的 JTextField 应该与作为输入的 JTextField 有所区别。将输出 JTextField 的可编辑属性设置为 false 可防止用户向 JTextField 中输入数据, 这同时也会使 JTextField 以灰色的背景出现在用户界面中。

Java 类库索引

JButton 该组件可允许用户命令应用程序完成一项操作。

- 运行



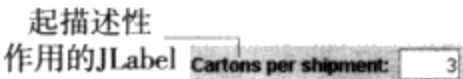
- 方法

setBounds 设置边界属性, 利用其指明 JButton 的位置与大小。

setText 设置 JButton 的文本属性。

JLabel 这个组件用来显示不允许用户进行修改的文本。

- 运行



- 方法

setBounds 设置边界属性, 利用其指明 JLabel 的位置与大小。

setFont 设置 JLabel 中所显示文本的字体名、字号及字形。例如, 为设置 textJLabel 的字体使用语句 textJLabel.setFont(new Font(fontName, fontStyle, fontSize)); 其中属于 fontName 的一些实例包括 "SansSerif", "Serif" 和 "Monospaced"。fontStyle 可以是 Font.PLAIN, Font.BOLD, Font.ITALIC 或 Font.Bold+Font.ITALIC。fontSize 可以是任意的正整数。

`setHorizontalAlignment` 用于指定 `JLabel` 内文本的对齐方式 (`JLabel.LEFT`, `JLabel.CENTER`, `JLabel.RIGHT`)。

`setIcon` 为 `JLabel` 设置显示的图片。

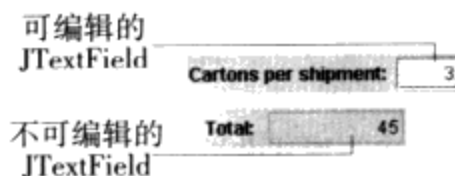
`setLocation` 设置 `JLabel` 的位置。

`setSize` 设置 `JLabel` 的宽度和高度 (以像素为单位)。

`setText` 设置 `JLabel` 中显示的文本。

TextField 该组件可以从键盘中获取输入或是将信息显示给用户。

● 运行



● 方法

`setBounds` 设置边界属性, 利用其指明 `TextField` 的位置与大小。

`setEditable` 用以指明用户是否可编辑 `TextField`。

`setHorizontalAlignment` 用以指明 `TextField` 内文本的对齐方式。

`setText` 用以指明 `TextField` 中所显示的文本。

习题

选择题

- 3.1 将 `TextField` 的 _____ 属性设置为 `false` 可防止用户对该 `TextField` 进行编辑。
a) 文本 b) 大小 c) 可编辑 d) `horizontalAlignment`
- 3.2 为对齐 `TextField` 中的文本, `horizontalAlignment` 属性可设置为 _____。
a) `TextField.Right` b) `TextField.RIGHT` c) `TextField.right` d) 以上答案均正确
- 3.3 在自定义 `JLabel` 时, 可指定该 `JLabel` 的 _____。
a) 文本对齐方式 b) 文本 c) 大小 d) 以上答案均正确
- 3.4 改变 _____ 属性值将改变 `TextField` 的大小和位置。
a) 文本 b) 大小 c) 范围 d) 位置
- 3.5 `TextField` 中文本的位置可通过 _____ 设置。
a) `setAlignmentProperty` b) `setAlignment` c) `setHorizontalAlignment` d) 以上答案均不对
- 3.6 可通过使用 _____ 帮助用户理解某个组件的意图。
a) `JButton` b) `TextField` c) `JLabel` d) 标题栏
- 3.7 使用 _____ 组件允许用户通过键盘输入数据。
a) `JButton` b) `TextField` c) `JLabel` d) 以上答案均不对
- 3.8 起描述性作用的 `JLabel` 应该使用 _____。
a) 语句大写形式 b) 书名大写形式 c) 一个冒号 (在文本的末尾) d) (a)和(c)
- 3.9 通过使用 _____ 可以设置 `JButton` 上的文本。
a) `setText` b) `setButtonText` c) `setJButtonText` d) `setText`
- 3.10 应该为 `JButton` 的文本属性使用 _____。
a) 书名大写形式 b) 语句大写形式 c) 一个冒号 (在文本的末尾) d) (a)和(c)

练习题

习题3.11至习题3.14要求读者针对每一道习题自定义一部分需要显示的GUI。读者应使用本教程中所讲授的GUI编程技术来自定义这些不同的GUI。由于只针对GUI的自定义, 因此这些应用程序并不能执行任何操作 (我们将在以后讲授如何实现这些应用程序的功能)。

- 3.11 (地址簿GUI)在这个习题中,读者需要将所学到的一些GUI设计规则应用到一个地址簿图形用户界面的设计中(如图3.18所示)。读者将设置 Address:JLabel 和 JTextField 的范围,使它们能够与其他 GUI 组件正确对齐。

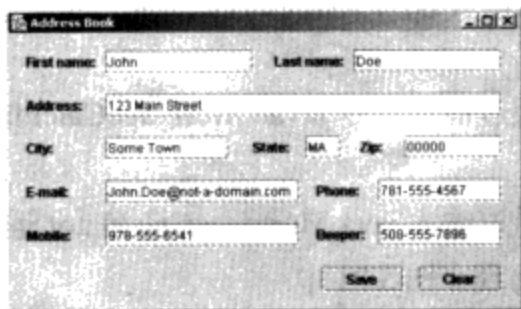


图 3.18 地址簿应用程序

- 将模板复制到工作目录中 将 C:\Examples\Tutorial03\Exercises\AddressBook 目录复制到 C:\SimplyJava 目录中。
- 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\AddressBook` 并按下 Enter 键进入到当前的工作目录中。
- 编译模板应用程序 键入命令 `javac AddressBook.java` 并按下 Enter 键,对该应用程序进行编译。
- 运行模板应用程序 通过键入 `java AddressBook` 运行此应用程序。地址簿模板应用程序的 GUI 将会出现在图 3.19 中。注意它与图 3.18 之间的不同。



图 3.19 地址簿模板应用程序

- 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
 - 打开模板文件 在自己的文本编辑器中打开 AddressBook.java 文件。
 - 自定义 Address:JLabel 和 JTextField 参照图 3.18 及模板代码,设置 addressJLabel 和 addressJTextField 的范围属性,使 addressJLabel 与 firstNameJLabel 靠左对齐(参见模板代码第 68 行),addressJTextField 与 firstNameJTextField 靠左对齐(参见模板代码第 74 行)。在第 91 行的下面,插入一条语句设置 addressJLabel 的范围属性,使其 y 坐标为 56,其 x 坐标、宽度和高度与 firstNameJLabel 的 x 坐标、宽度和高度相同。在第 97 行的下面,插入一条语句设置 addressJTextField 的范围属性,使其 y 坐标为 56,宽度为 360, x 坐标和高度与 firstNameJTextField 的 x 坐标和高度相同。
 - 保存应用程序 保存修改后的源代码文件。
 - 编译完成后的应用程序 在命令提示符窗口中,通过键入 `javac AddressBook.java` 并按下 Enter 键对应用程序进行编译。
 - 运行完成后的应用程序 若应用程序能够正确编译,通过输入 `java AddressBook` 运行它。将完成后的地址簿应用程序的 GUI 同图 3.18 中所示的 GUI 做一比较,以确信正确地自定义了 JLabel 和 JTextField。
 - 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。返回到命令提示符窗口下。
 - 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。
- 3.12 (抵押单据计算器GUI)在这个习题中,读者需要将所学到的一些GUI设计规则应用到一个抵押单据计算器的图形用户界面的设计中(如图3.20所示)。读者将设置 Loan amount:JLabel 和 JTextField 的范围,使它们能够与其他 GUI 组件正确对齐。

- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial03\Exercises\MortgageCalculator 目录复制到 C:\SimplyJava 目录中。
- b) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\MortgageCalculator` 并按下 Enter 键进入到当前的工作目录中。
- c) 编译模板应用程序 通过键入命令 `javac MortgageCalculator.java` 并按下 Enter 键, 对该应用程序进行编译。
- d) 运行模板应用程序 通过输入 `java MortgageCalculator` 运行此应用程序。抵押单据计算器模板应用程序的 GUI 将会出现在图 3.21 中。



图 3.20 抵押单据计算器应用程序



图 3.21 抵押单据计算器模板应用程序

- e) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- f) 打开模板文件 在自己的文本编辑器中打开 MortgageCalculator.java 文件。
- g) 自定义 Loan amount:JLabel 和 JTextField 参照图 3.20 及模板代码, 设置 loanAmountJLabel 和 loanAmountJTextField 的范围属性, 使 loanAmountJLabel 与 homeValueJLabel 靠左对齐 (参见模板代码第 57 行), loanAmountJTextField 与 homeValueJTextField 靠左对齐 (参见模板代码第 63 行)。在第 69 行的下面, 插入一条语句设置 loanAmountJLabel 的范围属性, 使其 y 坐标为 56, x 坐标、宽度和高度与 homeValueJLabel 的 x 坐标、宽度和高度相同。在第 75 行的下面, 插入一条语句设置 loanAmountJTextField 的范围属性, 使其 y 坐标为 56, x 坐标、宽度和高度与 homeValueJTextField 的 x 坐标、宽度和高度相同。
- h) 保存应用程序 保存修改后的源代码文件。
- i) 编译完成后的应用程序 在命令提示符窗口中, 通过键入 `javac MortgageCalculator.java` 并按下 Enter 键对应用程序进行编译。
- j) 运行完成后的应用程序 若应用程序能够正确编译, 通过输入 `java MortgageCalculator` 运行它。将完成后的抵押单据计算器应用程序的 GUI 同图 3.20 中所示的 GUI 做一比较, 以确信正确地自定义了 JLabel 和 JTextField。
- k) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。返回到命令提示符窗口下。
- l) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

3.13 (密码 GUI) 在这个习题中, 读者需要将所学习到的一些 GUI 设计规则应用到一个信息将受密码保护的图形用户界面的设计中 (如图 3.22 所示)。读者将设置 Enter your secret message:JTextArea 的范围, 使它的左侧能够与 Enter your secret message:JLabel 的左侧对齐而右侧能够与 Log In JButton 的右侧对齐。

- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial03\Exercises\Password 目录复制到 C:\SimplyJava 目录中。
- b) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\Password` 并按下 Enter 键进入到当前的工作目录中。
- c) 编译模板应用程序 通过键入命令 `javac Password.java` 并按下 Enter 键, 对该应用程序进行编译。

- d) **运行模板应用程序** 通过输入 `java Password` 运行此应用程序。密码模板应用程序的 GUI 将会出现在图 3.23 中。注意与图 3.22 之间的不同。

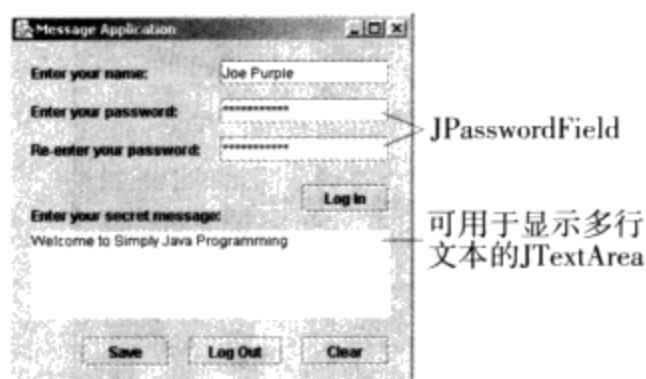


图 3.22 密码应用程序



图 3.23 密码模板应用程序

- e) **关闭应用程序** 点击关闭按钮关闭正在运行的应用程序。
- f) **打开模板文件** 在自己的文本编辑器中打开 `Password.java` 文件。
- g) **自定义 Enter your secret message:JTextArea** 参照图 3.22 及模板代码, 设置 `messageJTextArea` 的范围属性, 使 `messageJTextArea` 的左侧与 `messageJLabel` 的左侧对齐 (参见模板代码第 90 行), 右侧与 `logInJButton` 的右侧对齐 (参见模板代码第 84 行)。在第 95 行的下面, 插入一条语句设置 `messageJTextArea` 的范围属性, 使 `x` 坐标与 `messageJLabel` 的 `x` 坐标相同; `y` 坐标为 160; 根据 `messageJLabel` 和 `logInJButton` 的范围计算其宽度; 其高度为 72。
- h) **保存应用程序** 保存修改后的源代码文件。
- i) **编译完成后的应用程序** 在命令提示符窗口中, 通过键入 `javac Password.java` 并按下 `Enter` 键对应用程序进行编译。
- j) **运行完成后的应用程序** 若应用程序能正确编译, 通过输入 `java Password` 运行它。将完成以后的密码应用程序的 GUI 同图 3.22 中所示的 GUI 做一比较, 以确信正确地自定义了 `JTextArea`。本书将在后面讲授如何完整地实现应用程序的功能。
- k) **关闭应用程序** 点击关闭按钮关闭正在运行的应用程序。返回到命令提示符窗口下。
- l) **关闭命令提示符窗口** 点击命令提示符窗口的关闭按钮将其关闭。

挑战题

3.14 (计算机显示器发票 GUI) 在这个习题中, 读者需要将所学到的一些 GUI 设计规则应用到一个可开具发票的应用程序图形用户界面的设计中 (如图 3.24 所示)。读者将指定该 GUI 中以 15", 17" 和 19" 这些 `JLabel` 为起始行的一些 `JTextField` 的范围。

- a) **将模板复制到工作目录中** 将 `C:\Examples\Tutorial03\Exercises\MonitorInvoice` 目录复制到 `C:\SimplyJava` 目录中。
- b) **打开命令提示符窗口改变目录** 选择 `Start → Programs → Accessories → Command Prompt` 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\MonitorInvoice` 并按下 `Enter` 键进入到当前的工作目录中。
- c) **编译模板应用程序** 通过键入命令 `javac MonitorInvoice.java` 并按下 `Enter` 键, 对该应用程序进行编译。
- d) **运行模板应用程序** 通过键入 `java MonitorInvoice` 运行此应用程序。显示器发票模板应用程序的 GUI 将会出现在图 3.25 中。
- e) **关闭应用程序** 点击关闭按钮关闭正在运行的应用程序。
- f) **打开模板文件** 在自己的文本编辑器中打开 `MonitorInvoice.java` 文件。
- g) **自定义位于 Quantity:JLabel 之下的 JTextField** 参照图 3.24 及模板代码, 设置 `quantity15JTextField`, `quantity17JTextField` 和 `quantity19JTextField` 的范围属性, 使它们均与 `quantityJLabel` 一同靠左对齐 (参见模板代码 136 行) 并且分别与相应的 `JLabel`: `fifteenJLabel` (参见模板代码 154 行),

seventeenJLabel (参见模板代码 161 行) 和 nineteenJLabel (参见模板代码 168 行) 垂直对齐。这些 JTextField 的宽度和高度均分别为 80 和 21。

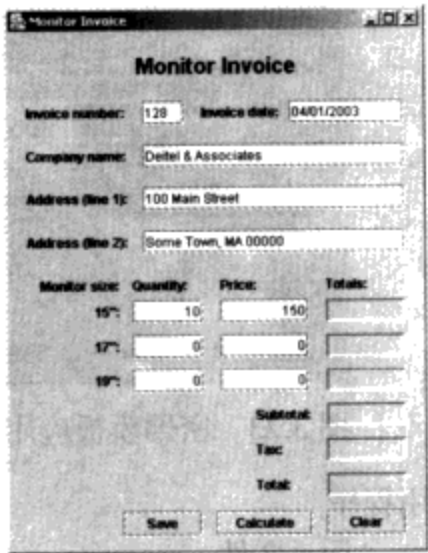


图 3.24 显示器发票应用程序

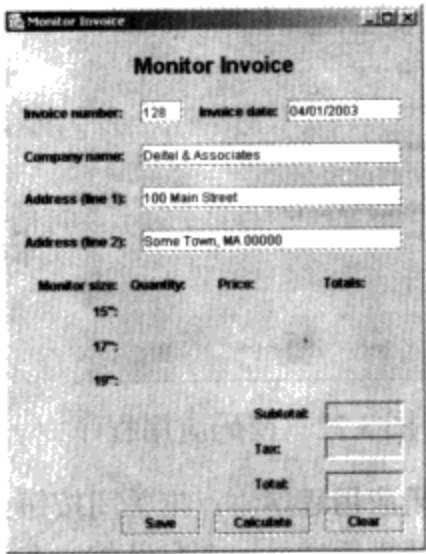


图 3.25 显示器发票模板应用程序

- h) 自定义位于 Price:JLabel 之下的 JTextField 参照图 3.24 及模板代码, 设置 price15JTextField, price17JTextField 和 price19JTextField 的范围属性, 使它们均与 priceJLabel 一同靠左对齐 (参见模板代码 142 行) 并且分别与相应的 JLabel:fifteenJLabel (见模板代码 154 行), seventeenJLabel (参见模板代码 161 行) 和 nineteenJLabel (参见模板代码 168 行) 垂直对齐。这些 JTextField 的宽度和高度均分别为 80 和 21。
- i) 自定义位于 Total:JLabel 之下的 JTextField 参照图 3.24 及模板代码, 设置 totals15JTextField, totals17JTextField 和 totals19JTextField 的范围属性, 使它们均与 totalJLabel 一同靠左对齐 (见模板代码 187 行) 并且分别与相应的 JLabel:fifteenJLabel (参见模板代码 154 行), seventeenJLabel (参见模板代码 161 行) 和 nineteenJLabel (参见模板代码 168 行) 垂直对齐。这些 JTextField 的宽度和高度均分别为 80 和 21。
- j) 保存应用程序 保存修改后的源代码文件。
- k) 编译完成后的应用程序 在命令提示符窗口中, 通过键入 javac MonitorInvoice.java 并按下 Enter 键对应用程序进行编译。
- l) 运行完成后的应用程序 若应用程序能正确编译, 通过输入 java MonitorInvoice 运行它。将完成后的显示器发票应用程序的 GUI 同图 3.24 中所示的 GUI 做一比较, 以确信正确地自定义了这些 JTextField。
- m) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- n) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。



教程 4 完整的库存清单应用程序

引入程序设计的概念

教学目标

在本教程中，读者将学到以下内容：

- 如何使应用程序执行操作以响应对 JButton 的点击操作
- 如何正确使用乘法运算符
- 利用 Integer.parseInt 方法将字符串转换为整数
- 利用 String.valueOf 方法将数值转换为字符串

在本教程中，将引入用户交互式应用程序在创建过程中可能涉及到的一些与 Java 编程技术相关的基础知识。当读者将相关功能（通过 Java 代码）添加到教程 3 中曾设计的库存清单应用程序中时，将学习一些编程概念。所谓功能，描述的是应用程序可用于执行的任务。在本教程中，读者还将考察代表诸如点击某个 JButton 或者改变 JTextField 中的值的一些用户操作事件，以及当这样的事件发生时，作为执行（调用）的代码段的事件处理程序。最后，读者将了解到事件及其事件处理程序对于编写 GUI 应用程序的重要性。

4.1 探试库存清单应用程序

在本教程中，读者将完成从教程 3 中就开始的库存清单应用程序。下面首先回忆一下该应用程序必须满足的需求：

应用程序需求分析

某大学书店接收到几箱教材。在一批教材中，每箱教材的数量是相同的。仓库管理员希望使用一台计算机来计算书店收到的每批教材的总数量。仓库管理员会输入一批教材中箱子的数目及每只箱子中所存放教材的固定数目；然后，该应用程序会计算并显示出这批教材的总数量。

库存清单管理员已审阅并通过了前面的设计。现在需要添加一些代码，使用户在点击一个 JButton 时，应用程序能够把箱子的数目与每箱教材的数目进行一个乘法运算并显示出计算的结果。即得到所收到教材的总数目。我们将从这个完成后的应用程序作为探试的开始。随后，学习一些 Java 技术，作为创建一个属于自己应用程序的基础。



探试完成后的库存清单应用程序

1. 定位到完成后的应用程序 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\Examples\Tutorial04\CompletedApplication\Inventory2` 然后按下 Enter 键，将目录改变到这个完成后的库存清单应用程序的目录下面（注意：从现在起，凡在命令提示符窗口下所输入的每条命令的后面，将不再提示读者按下 Enter 键）。

- 2. 运行库存清单应用程序 在命令提示符窗口下键入 java Inventory 运行该应用程序。在 Cartons per shipment:JTextField 中键入 3，在 Items per carton:JTextField 中键入 15（如图 4.1 所示）。
- 3. 计算所收到的货物总数 点击 Calculate Total 这一 JButton。应用程序将对输入的两个数做乘法运算并在 Total:JTextField 中（如图 4.2 所示）显示出计算的结果（45）。

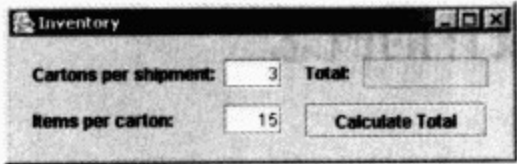
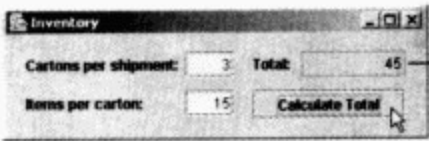


图 4.1 输入数值后的库存清单应用程序



— 计算的结果

图 4.2 在库存清单应用程序中计算总数量

- 4. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- 5. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

4.2 介绍 Java 的代码规则

在教程 2 和教程 3 中，学习了一些基本的 Java 语句，其主要作用是自定义应用程序的 GUI。在这一节里，将学习 Java 编程技术的一些其他技术并利用这些新的知识对曾经创建的那个库存清单应用程序进行改进。

众所周知，GUI 程序设计是比较有趣的，使用它能够创建出吸引人的可视化应用程序，但是，GUI 程序设计本身并不足以完成整个应用程序的全部。到目前为止，作为开发这个库存清单应用程序的程序员，我们已经完成了一些如设置 JLabel 中的文本或者是设置 JButton 位置的工作。然而，这个应用程序还并未完成一些基于用户输入的操作。比如说，开发人员需要通过编程才可以完成诸如点击 JButton 等的操作。而完成这一类问题的关键在于能否正确地将 GUI 组件同每个应用程序的功能融合起来。在下面“介绍 Java 的代码规则”中，读者将首先了解如何使用 Java 代码为应用程序添加所需要的功能。

介绍 Java 的代码规则

- 1. 将模板复制到工作目录中 将 C:\Examples\Tutorial04\TemplateApplication\Inventory2 目录复制到 C:\SimplyJava 目录中。该目录中含有一个曾在教程 3 中所创建的应用程序作为基础模板。
- 2. 打开库存清单应用程序的模板文件 在文本编辑器中打开模板文件 Inventory.java（如图 4.3 所示）。



图 4.3 利用文本编辑器对库存清单应用程序的一部分代码进行显示

所有的 Java 应用程序均是由一些称为类的部分组成的，它们简化了应用程序的组织形式。在这些类中，包含了一些称为方法的部分。方法中含有能执行特定任务的一些语句，而这些语句又是由代码组成的

(通常将多条语句组成的代码组称为代码块),并且这些方法常常会在任务完成时返回相应的信息。本例中的这个模板代码里面定义了该库存清单应用程序所使用到的一些类(图4.3中从第8行至第118行总共被称为一个类的定义)。大多数的Java应用程序是由程序员编写的代码、Sun Microsystems编写并提供的一些预先存在的类,以及其他在Java API中的类等几部分组合而成的。成功开发Java应用程序的关键是能够正确地将这几个方面融合在一起。我们将学习如何在应用程序中使用这两种技术。



好的编程习惯

类标识符中的每一个单词的首字母应采取大写形式,如标识符 SampleClassName。

3. 考察类的定义 从图4.3中的第8行开始,定义了一个类。关键字 class 在Java中表明引入了一个类的定义,紧接其后会跟有一个类名(此应用程序中为 Inventory)。Java中的每一个应用程序至少是由程序员定义的某个类所组成的。按照惯例,Java中所有类的类名都是以一个大写字母作为起始的,并且类名中每一个新单词的第一个字母也应作为大写(如 SampleClassName)。类名是通过标识符来指定的。标识符是指由字母、数字、下划线(_)和美元符号(\$)所组成的一个字符序列。标识符不能以数字作为起始且不包含空格。一些属于有效标识符的例子有: Welcome1, \$value1, label_Value, exitJButton 和 _total。字符序列 7welcome 并不是一个有效的标识符,因为它以一个数字作为起始,而 input field 也不是一个有效的标识符,因为含有一个空格。回顾我们曾经说过的,Java是区分大小写的——也就是说,大写和小写字母是不同对待的。因此, a1 和 A1 是两个不同的标识符(都是合法的)。



常见编程错误

在Java关键字中采用大写的字母会导致应用程序编译时的语法错误。

关键字(或称保留字)是保留给Java使用的,因此,决不能把关键字作为自己的标识符来使用。附录E中有一个Java关键字的完整列表。读者将在本书中学习到绝大多数的Java关键字。

当类的声明在一个文件中保存时,文件名必须是由一个类名后跟一个.java的扩展名所组成的。如本例中的这一应用程序的文件名为 Inventory.java。所有Java类的声明都保存在以.java作为文件扩展名的文件中。左花括号(参见图4.3中第9行),“{”,是每一个类定义的开始,而相应的右花括号(参见模板中第118行),“}”,则代表每一个类定义的结束。这里的这个库存清单应用程序的代码是由 Inventory 类的类体组成的。我们还将在本教程和教程5中讨论更多的位于模板代码里的元素。



常见编程错误

在类的声明中省略任何的花括号会导致出现语法错误。

4. 对 JFrame 类继承的理解 每一个 GUI 应用程序是由至少一个继承 Java API (Java 中已经定义好的类库) 中的 JFrame (参见图4.3第8行) 的类组成的。关键字 extends 表明类 Inventory 继承了另一个类的成员(这通常称为扩展一个类)。利用继承机制扩展 JFrame, 应用程序便可以将类 JFrame 作为一个“模板”来使用。从类 JFrame 中继承, 能够带来的一个关键性的好处是 Java API 中已经定义好了“JFrame 的含义”。读者在屏幕中所看到的这个窗口实际已具备了一定的功能。这样, 由于类 JFrame 已经提供了这些功能, 程序员就不必再次定义这些功能了, 因为根本无需去做“重新发明轮子”的尝试。扩展类 JFrame 能够让开发人员更快、更方便地创建 GUI。

在以上示例中, 我们考察了模板代码中类的声明。利用类声明中的这些代码, 将在 JFrame 中创建并自定义应用程序执行时出现的相关 GUI 组件。

自测题

1. 标识符 _____。

- | | |
|-------------------------|-----------------------|
| a) 可以以任意字符作为起始, 但不能包含空格 | b) 必须以数字作为起始, 但不能包含空格 |
| c) 不能以数字作为起始, 也不能包含空格 | d) 不能以数字作为起始, 但可以包含空格 |

2. 关键字 _____ 表示某个新类是从一个现有的类中继承的。
- a) inherits b) extends c) reuses d) 以上答案均不对

答案：1) c 2) b

4.3 在事件处理程序中放置代码

接下来，我们将修改这一应用程序使之能对用户的输入做出响应。本书中绝大多数的 Java 应用程序是按照事件处理程序的方式来提供相应功能的。回顾曾经提到过内容，当某个事件产生时，例如，点击一个 JButton，便会执行一个事件处理程序。

查看 JButton 的 actionPerformed 事件处理程序

1. 查看事件处理程序 我们为读者提供了本应用程序的一个事件处理程序的模板。图 4.4 中的第 105 行至第 108 行显示的是 calculateJButton 的 actionPerformed 事件的事件处理程序。
- 这个事件处理程序包含了库存清单应用程序的用户点击 Calculate Total JButton 时所需执行的代码。更确切地说，点击 Calculate Total JButton 会产生一个 actionPerformed 事件，反过来将导致 105 行至第 108 行这个事件处理程序的执行。

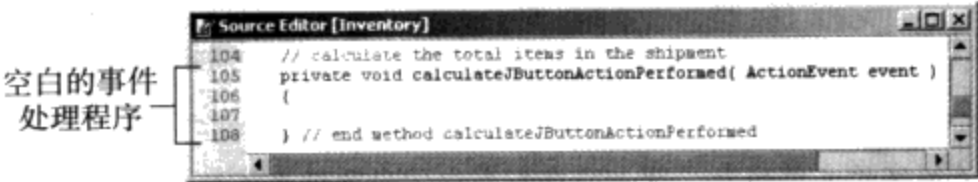


图 4.4 添加应用程序所需代码之前的空事件处理程序 calculateJButtonActionPerformed

- 注意 calculateJButtonActionPerformed 这个名字用于响应 Calculate Total JButton 的 actionPerformed 事件的事件处理程序。这里为事件处理程序而采用的一种命名约定是：将 GUI 组件名 (calculateJButton) 同用户与该组件交互时所发生的事件 (actionPerformed) 相互结合在了一起。采用这种约定可以增强程序的可读性，并且能够将事件处理程序同相应的 GUI 组件匹配起来。例如，如果应用程序中包含了一个名为 submitJButton 的 JButton，其事件处理程序则为 submitJButtonActionPerformed。
2. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\Inventory2 进入到当前的工作目录中。
3. 编译应用程序 通过键入命令 javac Inventory.java，对该应用程序进行编译。
4. 运行应用程序 通过键入 java Inventory 运行此应用程序。图 4.5 中显示的是运行时的应用程序。可以点击 Calculate Total JButton。注意到，尽管代码中包含了 Calculate Total JButton 的 actionPerformed 事件的一个事件处理程序，但点击 Calculate Total JButton 时并无任何的操作发生，这是因为我们尚未给事件处理程序添加任何代码。在下一个示例中，将添加用户点击 Calculate Total JButton 时所执行的代码。该代码将计算出一批货物的总数量。

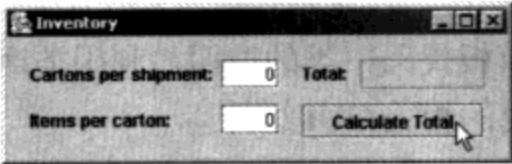


图 4.5 在为事件处理程序添加功能之前运行该应用程序

5. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
6. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

既然已经查看过了 Calculate Total JButton 事件处理程序的模板代码，下面读者将在该事件处理程序中添加一些代码，以便用户能够在点击 Calculate Total JButton 时，对两个整数进行相乘运算并把结果显示在 Total:JTextField(totalResultJTextField) 中。这样做的目的是，为了能够更好地说明某个数值结果是如何通过一个 JTextField 来进行显示的，下列示例中将一直采用整数 3 和整数 15 来完成这个计算，也就是说，不考虑用户在 Cartons per shipment: 和 Items per carton:JTextField 中输入的值。因此，其计算结果将总为 45。在完成了下列示例中的任务以后，再去插入可用于完成正确计算的代码，最终实现这个库存清单应用程序。

向空白事件处理程序中添加代码

1. 打开库存清单应用程序的模板文件 在文本编辑器中打开模板文件 Inventory.java。
2. 往事件处理程序中添加代码 将屏幕滚动至 JButton 事件处理程序所在的第 105 行，并插入图 4.6 中的第 107 行至第 108 行。

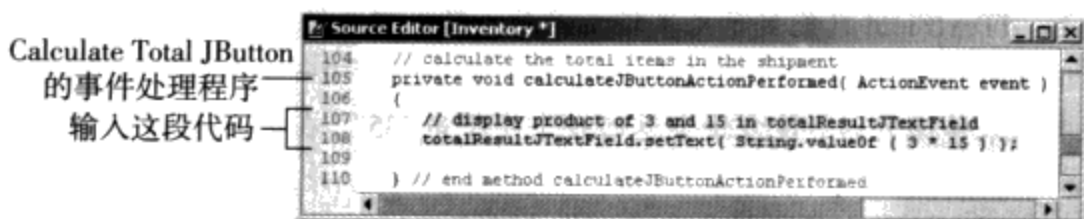


图 4.6 往 Calculate Total JButton 事件处理程序中添加代码

图 4.6 中的第 107 行是以两个正斜杠字符 (//) 作为起始的，这表明本行将是一个注释。在应用程序中添加注释可以改善代码的可读性。由于注释可对代码进行解释，因此，开发人员（以及其他可能的程序员）便能够更容易地理解自己的代码。注释还有助于日后对这些代码重新进行理解。



好的编程习惯

在代码行的末尾处添加注释时，注意之前应留有一个或更多空格，以提高程序的可读性。



好的编程习惯

为整行都作为注释的注释行的上面放置一个空白行。这一空白行能够使该注释更加突出并且提高了程序的可读性。



常见编程错误

若在语句的末尾处忘记添加分号，会导致应用程序编译错误。

可将注释放置在独立的一行中（它们被称做“整行注释”）或者是放置在某行代码的末尾处（它们被称为“行末注释”）。在许多文本编辑器中，注释是以绿色的形式出现的。

Java 编译程序会忽略掉这些注释，因此在应用程序运行时，它们并不会引发计算机去执行任何的操作。

第 107 行中的这个注释仅仅是描述了第 108 行中的语句所能完成的任务。注意图 4.6 中第 110 行里的注释，该行中出现的这个封闭的花括号 (})，表示了事件处理程序的结束。

读者或许记得曾经在教程 2 和教程 3 中看到过类似第 108 行中的这条语句，有关这一代码的详细内容将随后进行讲解。其实，图 4.6 中的第 108 行是一条可完成某项操作的可执行语句。而每一条语句都是以一个分号 (;) 字符作为结束的（比如第 108 行）。这条语句将用来设置 totalResultJTextField 的文本属性。下面，让我们更加详细地来了解这条语句。通常，为了设置某个组件的文本属性，需使用 setText 方法。例如，通过编写：

```
componentName.setText("text to set");
```

会使某个称之为 componentName 的组件在屏幕上显示出文本字符串 "text to set"。再比如，当执行到第 198 行的这条语句：

```
totalResultJTextField.setText( "100" );
```

时, 会把 totalResultJTextField 的文本属性设置为 "100", 然后再显示到 JTextField 之中。

上面使用的是一种称为方法 (setText) 的语句来对 totalResultJTextField 的文本属性值执行修改操作的。方法是指一段能够在调用时 (执行时) 执行某项任务或操作的代码。有时, 方法会将某个值返回到调用它们的位置处。调用一个方法是通过输入其名且后跟一对圆括号来完成的。圆括号中的任何值 (例如, 在前面语句中的 "100") 均属于方法的参数。传递至方法的参数提供了该方法用于完成某项任务的相关信息。如方法 setText 需要有一个参数, 用做指定希望修改某个组件的文本属性值。我们将在教程 12 中学习如何创建自己的方法。

访问某个组件的属性是通过指定该组件的名称后跟一个点 (.) 最后再加上用于访问该属性的方法的名称来完成的。这个点有时也被认为是点分隔符。

在第 108 行, setText 方法的参数是:

```
String.valueOf( 3 * 15 )
```

所调用的 String.valueOf 方法将一个数值转换成文本, 而其中所执行的计算则是通过使用数值来完成的。遗憾的是, JTextField 只能显示文本值。因而, 数值需要转换成文本才可用于组件文本属性的设置。String.valueOf 方法将所提供的数据作为它的参数并且返回一个相应的文本值。例如, 如果将数值 35 作为一个参数来传递的话, 该方法会返回一个字符串 "35"。在第 108 行, 由 String.valueOf 方法返回的字符串将成为 totalResultJTextField 的 setText 方法的参数, 并最终显示在 totalResultJTextField 中。第 108 行中的 String.valueOf 方法的参数是一个表达式 $3 * 15$ 。星号 (*) 被认为是乘法运算符, 它对两个数值执行相乘运算并返回它们的乘积。在代数运算中, 通常是使用一个位于中间的点运算符作为表示乘法运算的运算符, 像 $3 \cdot 15$ 。然而, 处于中间的点运算符在计算机键盘上并不可用, 因此大多数的编程语言用星号 (*) 字符来替代。乘法运算符任意一侧的表达式被称为操作数。该运算符是将运算符左侧的值 (左操作数) 与运算符右侧的值 (右操作数) 进行相乘运算。乘法运算符被认为是一种二目运算符, 因为它需要有两个操作数。在这个例子中, 操作数分别是整数 3 和整数 15 (注意: 读者还可以使用 +, - 和 / 来分别执行加法、减法和除法的运算)。

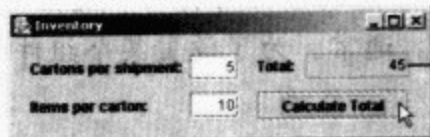
当用户点击 Calculate Total JButton 时, 第 105 行至第 110 行 (如图 4.6 所示) 的事件处理程序将开始执行, 即显示出表达式 $3 * 15$ (即 45) 的结果。很显然, 针对用户在 Cartons per shipment: JTextField 和 Items per carton: JTextField 中输入的任意值, 的确得不到一个正确的结果——正确的结果应该是每只箱子中货物的数目乘以每批货物中箱子的数目。在下一个示例中, 将学习如何对这一错误进行改正。

3. 保存应用程序 保存修改后的源代码文件。

4. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\Inventory2` 进入到当前工作目录中。

5. 编译应用程序 通过键入命令 `javac Inventory.java`, 对该应用程序进行编译。如果没有编译错误, 进入第 6 步。否则, 改正代码中的错误并重复本步骤。

6. 运行应用程序 若此应用程序能够正确编译, 可以通过键入 `java Inventory` 来运行它。图 4.7 显示了运行时的应用程序。在分别向 Cartons per shipment: JTextField 和 Items per carton: JTextField 中键入 5 和 10 以后, 按下 Calculate Total JButton。注意 totalResultJTextField 中仍旧显示的是 45, 这主要是依据第 108 行中完成的计算而得出的结果。同样, 读者将在下一节中利用 Cartons per shipment: JTextField 和 Items per carton: JTextField 中实际输入值的计算来改正这个错误。



点击 Calculate Total JButton 时的结果

图 4.7 利用事件处理程序执行应用程序

7. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。

8. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

自测题

- 1. 方法 _____ 可将一个数值转换成文本。
a) Integer.valueOf b) Integer.parseInt c) String.parseInt d) String.valueOf
- 2. 乘法运算符一侧的表达式被引述为它的 _____。
a) 运算符的值 b) 结果 c) 操作数 d) 参数

答案: 1) d 2) c

4.4 执行计算并显示结果

现在我们已经熟悉了如何在JTextField中显示输出，以下示例将通过计算并显示一批货物中箱子数目与每只箱子中货物数目的乘积来完成这个库存清单应用程序。

完成库存清单应用程序

- 1. 打开库存清单应用程序的模板文件 在文本编辑器中打开模板文件 Inventory.java。
- 2. 添加一条多行语句 将屏幕滚动至图 4.6 中第 105 行 calculateJButton 事件处理程序所在的位置处。将属于 calculateJButtonActionPerformed 的第 107 行至第 108 行替换成图 4.8 中的第 107 行至第 110 行。回顾在前面“向空白事件处理程序中添加代码”的示例中，一条 Java 语句是利用一个分号作为结束的。注意第 108 行至第 110 行中只有第 110 行的行末有一个分号。这种情况下，第 108 行至第 110 行表示的是一条多行语句。也就是说一条语句可以跨越到多行上，因为 Java 会忽略代码中的额外空格、制表符及换行符。换行符是当按下 Enter 键时被插入的字符。空格、制表符及换行符统称为空白（注意：空白在由字符串文字所组成的双引号中是不能被忽略的部分）。程序员通常会将较长的语句分割成多行以增强代码的可读性。同样，注意到该语句所在的第 109 行至第 110 行进行了适当的缩进处理。利用缩进处理能够从视觉上表明该语句紧接着前一条语句。

从cartonsJTextField和itemsJTextField中读取数值，将它们转换成整数并做相乘运算，最后将结果显示在totalResultJTextField中

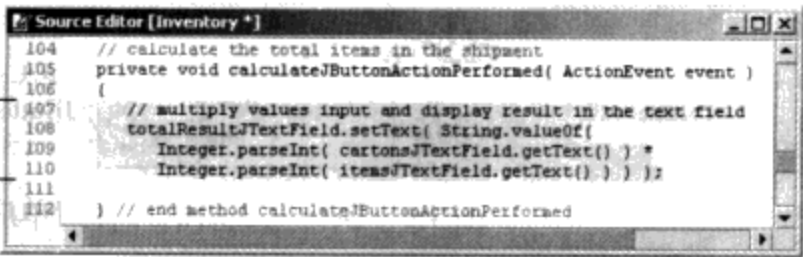


图 4.8 在库存清单应用程序中使用乘法运算



好的编程习惯

较长的一条语句可以延伸至多行上。若单个一条语句必须被分解至多行上时，应考虑选择一些有意义的断开点，比如某个运算符的后面。另外，如果一条语句被分解至两个或更多的行上时，通过某种缩进“级别”对所有的后续行进行缩进处理。



常见编程错误

在字符串的中间位置处断开一条语句将导致一个语法错误。

第 108 行至第 110 行将完成有关的乘法计算并且使用 setText 设置 totalResultJTextField 的文本属性。同前一个框图向空白事件处理程序中添加代码一样，传递给 setText 的参数是调用 String.valueOf 方法的结果，方法 String.valueOf 会将一个数值转换成文本。

在第 108 行至第 110 行里，String.valueOf 方法会接收下列 Java 代码的结果并将其作为它的参数来使用：

```
Integer.parseInt( cartonsJTextField.getText() ) *
Integer.parseInt( itemsJTextField.getText() )
```


这段代码是将来自 Cartons per shipment: JTextField 和 Items per carton:JTextField 的两个值执行乘法运算。正如数值需要转换成文本才可以显示一样,从 JTextField 中读取的文本必须被转换成数值以后才可以用于计算(如乘法运算)。第 109 行和第 110 行均调用了可用于转换的方法 Integer.parseInt,它是将一个文本值作为其参数并返回一个等价的整数值。这些文本值均来自于 Cartons per shipment: JTextField 和 Items per carton:JTextField 中的文本属性值。

还可以通过使用 getText 方法获取某个组件的文本属性,如:

```
componentName.getText()
```

将获得某组件在屏幕上所显示的文本。在库存清单应用程序中,若向图 4.7 中的 Cartons per shipment: JTextField 中键入 5,则图 4.8 中第 109 行的 cartonsJTextField.getText()将获得 Cartons per shipment: JTextField 中的这一文本属性值,其结果是一个包含 5 的字符串。由于 cartonsJTextField.getText()出现在 Integer.parseInt 方法调用的括号里,调用 cartonsJTextField.getText()的结果(即包含 5 的字符串)将被用做 Integer.parseInt 方法的参数。Cartons per shipment:JTextField 文本属性值中的文本被转换为乘法运算中作为左操作数的一个整数。类似地,在第 110 行,itemsJTextField.getText()的结果将作为参数传递给 Integer.parseInt,以便得到乘法运算中作为右操作数来使用的每箱货物的数目。

在使用 Integer.parseInt 时,读者需多加小心。如果传递给该方法中的文本并不是一个代表整数的值,那么会有一个异常产生。Java 是利用异常来表示应用程序在执行期间可能出现的问题。默认情况下,异常信息将输出到命令提示符窗口中。将在教程 24 中学习如何编写异常处理的代码。

3. 保存应用程序 保存修改后的源代码文件。

4. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\Inventory2 进入到当前的工作目录中。

5. 编译应用程序 通过键入命令 javac Inventory.java,对该应用程序进行编译。如果没有编译错误,进入第 6 步。否则,改正代码中的错误并重复本步骤。

6. 运行应用程序 若此应用程序能正确编译,可以通过键入 java Inventory 来运行它。图 4.9 显示了运行时的应用程序。现在,若在两个 JTextField 中输入数据(分别输入 5 和 10)并点击 Calculate Total JButton 的话,应用程序能够正确地计算出这两个数的乘积并把结果(50)显示在 totalResultJTextField 中。

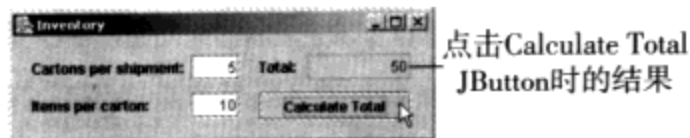


图 4.9 执行完成后的库存清单应用程序

7. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。

8. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

图 4.10 中给出了库存清单应用程序的源代码。本教程中,凡需要添加、查看或是修改的代码均在图中相应的代码行中做了突出显示。

```
1 // Tutorial 4: Inventory.java
2 // Calculates the number of items in a shipment based on the number
3 // of cartons received and the number of items per carton.
4 import java.awt.*;
5 import java.awt.event.*;
6 import javax.swing.*;
7
8 public class Inventory extends JFrame
9 {
10     // JLabel and JTextField for cartons per shipment
11     private JLabel cartonsJLabel;
12     private JTextField cartonsJTextField;
```

```
13
14 // JLabel and JTextField for items per carton
15 private JLabel itemsJLabel;
16 private JTextField itemsJTextField;
17
18 // JLabel and JTextField for total items per shipment
19 private JLabel totalJLabel;
20 private JTextField totalResultJTextField;
21
22 // JButton to initiate calculation of total items per shipment
23 private JButton calculateJButton;
24
25 // no-argument constructor
26 public Inventory()
27 {
28     createUserInterface();
29 }
30
31 // create and position GUI components; register event handlers
32 public void createUserInterface()
33 {
34     // get content pane and set layout to null
35     Container contentPane = getContentPane();
36     contentPane.setLayout( null );
37
38     // set up cartonsJLabel
39     cartonsJLabel = new JLabel();
40     cartonsJLabel.setText( "Cartons per shipment:" );
41     cartonsJLabel.setBounds( 16 , 16 , 130, 21 );
42     contentPane.add( cartonsJLabel );
43
44     // set up itemsJLabel
45     itemsJLabel = new JLabel();
46     itemsJLabel.setText( "Items per carton:" );
47     itemsJLabel.setBounds( 16 , 48, 104 , 21 );
48     contentPane.add( itemsJLabel );
49
50     // set up totalJLabel
51     totalJLabel = new JLabel();
52     totalJLabel.setText( "Total:" );
53     totalJLabel.setBounds( 204 , 16 , 40 , 21 );
54     contentPane.add( totalJLabel );
55
56     // set up cartonsJTextField
57     cartonsJTextField = new JTextField();
58     cartonsJTextField.setText( "0" );
59     cartonsJTextField.setBounds( 148, 16 , 40 , 21 );
60     cartonsJTextField.setHorizontalAlignment( JTextField.RIGHT );
61     contentPane.add( cartonsJTextField );
62
63     // set up itemsJTextField
64     itemsJTextField = new JTextField();
65     itemsJTextField.setText( "0" );
66     itemsJTextField.setBounds( 148, 48, 40, 21 );
67     itemsJTextField.setHorizontalAlignment( JTextField.RIGHT );
68     contentPane.add( itemsJTextField );
69
70     // set up totalResultJTextField
71     totalResultJTextField = new JTextField();
72     totalResultJTextField.setBounds( 244, 16 , 86 , 21 );
73     totalResultJTextField.setHorizontalAlignment(
74         JTextField.RIGHT );
75     totalResultJTextField.setEditable( false );
```

```

76     contentPane.add( totalResultJTextField );
77
78     // set up calculateJButton
79     calculateJButton = new JButton();
80     calculateJButton.setText( "Calculate Total" );
81     calculateJButton.setBounds( 204 , 48, 126, 24 );
82     contentPane.add( calculateJButton );
83     calculateJButton.addActionListener(
84
85         new ActionListener() // anonymous inner class
86         {
87             // event handler called when calculateJButton is pressed
88             public void actionPerformed((ActionEvent event) )
89             {
90                 calculateJButtonActionPerformed( event );
91             }
92
93         } // end anonymous inner class
94
95     ); // end call to addActionListener
96
97     // set properties of application's window
98     setTitle( "Inventory" ); // set title bar text
99     setSize( 354, 112 ); // set window size
100    setVisible( true ); // display window
101
102 } // end method createUserInterface
103
104 // calculate the total items in the shipment
105 private void calculateJButtonActionPerformed( ActionEvent event )
106 {
107     // multiply values input and display result in the text field
108     totalResultJTextField.setText( String.valueOf(
109         Integer.parseInt( cartonsJTextField.getText() ) *
110         Integer.parseInt( itemsJTextField.getText() ) ) );
111
112 } // end method calculateJButtonActionPerformed
113
114 // main method
115 public static void main( String[] args )
116 {
117     Inventory application = new Inventory();
118     application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
119
120 } // end method main
121
122 } // end class Inventory

```

从 cartonsJTextField
和 itemsJTextField 中
读取数值，将它们转
换成整数，并做相乘
运算，最后将结果显示
在 totalResultJTextField 中

图 4.10 库存清单应用程序代码

自测题

1. 方法 Integer.parseInt_____。

a) 将一个整数转换成文本

b) 将文本转换为一个整数

c) 同 String.valueOf 一样完成相同的任务

d) 以上答案均不对

2. 若 totalJTextField 是一个 JTextField，可以使用_____获取 totalJTextField 的文本属性值。

a) totalJTextField.getText()

b) totalJTextField.setText()

c) totalJTextField.getText()

d) totalJTextField.text()

答案：1) b 2) c

4.5 小结

在本教程中，读者学习了如何使用一个 `JTextField` 组件让用户输入数据以及如何使用一个 `JButton` 组件告诉正在运行的应用程序完成某个特定的操作。同时，还看到了一些在类的声明中所使用的可用于创建某个应用程序 GUI 的代码。尽管现在还不期望读者能完全地理解这些代码，但是应该能够认识到一个有关养成良好编程风格的关键因素，这就是在部署 GUI 组件和编写相关 GUI 组件功能的代码之间如何获得正确的平衡。

在大致了解了 Java 中的方法和运算符以后，通过在一个事件处理程序中插入一些代码，完成了某个简单的乘法运算并将最终结果显示给了用户。同时，读者还使用到了一些用于提高程序代码的可读性的注释。经过这些步骤，读者了解到通过在事件处理程序中放置一些代码，便可允许应用程序响应一个如点击某 `JButton` 的事件。

在下一个教程中，还将通过创建一些能使应用程序存储日后将使用的相关信息的变量，继续对这个库存清单应用程序进行改进。其中的一个改进会使用到 `keyPressed` 事件，这种事件是在用户改变 `JTextField` 中的值时发生的。在应用过这些变量的知识以后，将利用一个应用程序运行时的调试程序，删除应用程序中的某个逻辑错误。

技术小结

访问组件的属性值

- 将修改或访问某种值方法的方法名置于组件名和点分隔符(.)之后。例如，要取得名为 `cartonsJTextField` 的 `JTextField` 的文本属性，使用 `cartonsJTextField.getText()`，而要设置 `JTextField` 的文本属性，则使用 `cartonsJTextField.setText("value")`。

在代码中插入注释

- 利用两个正斜杠字符(//)作为注释的开始，然后就可以插入描述代码作用的文本了，其目的是更好地理解该段代码。可以将注释放置在单独的一行上或者是某行的末尾处。

为事件处理程序命名

- 使用格式 `componentNameEventName` 为某个事件处理程序命名，当中的 `componentName` 将作为与某事件相联系的组件的名字，而 `eventName` 则是该事件的名字。例如，`calculateJButtonActionPerformed` 将作为 `calculateJButton` 的 `actionPerformed` 事件的事件处理程序。

使用乘法运算符

- 在两个做乘法运算的数值操作数之间使用一个星号(*)。乘法运算符会将其左右的两个操作数执行乘法运算（注意：还可以使用 +、- 和 / 分别执行加法、减法和除法运算）。

从 JTextField 中取得一个整数值并将其转换成一个整数

- 利用方法 `getText()` 访问 `JTextField` 的文本属性，然后将结果作为方法 `Integer.parseInt` 的参数。例如，下列表达式是将 `itemsJTextField` 中的文本转换成一个整数：

```
Integer.parseInt(itemsJTextField.getText())
```

将数值转换成一个可在 JTextField 中显示的文本

- 提供一个数值作为方法 `String.valueOf` 的参数，该方法将返回一个包含等价文本值的字符串。然后，再将结果传递给 `JTextField` 的 `setText` 方法。例如，下列语句可将数 100 转换成一个文本值，并在 `totalResultsJTextField` 中显示出这一文本：

```
totalResultJTextField.setText(String.valueOf(100));
```


关键术语

actionPerformed 事件 当点击 JButton 时发生的一种事件。

参数 一个传递给方法的值，具体做法是：在某方法调用时，将所需要的一些值放置在紧跟方法名之后的括号内。

语句块 封闭在花括号内的一组语句。

二目运算符 一种需要具备两个操作数的运算符。

区分大小写 具备相同拼写形式的标识符，如果标识符的大小写不同，则会被区别对待。

类的声明 定义某个类的代码，以 class 关键字作为起始。

class 关键字 作为某类声明开始的关键字。

注释(//) 为提高应用程序的可读性而插入的解释性文本。

点分隔符 允许程序员调用某个特定的类或对象的方法。

行末注释 出现在某代码行末尾的注释。

事件 一种用于激发某事件处理程序的操作。

事件处理程序 当某个特定事件发生时所执行的代码。

扩展某个类 创建一个基于某现有类的新类（也称为继承）的机制。

extends 关键字 用于指明某类将从一个现有类中继承数据和功能的一个关键字。

正斜杠字符 以两个正斜杠(//)作为起始的注释。

整行注释 源代码中作为一个整行出现的注释。

功能 应用程序能够执行的任务或操作。

getText 方法 访问（或取得）某个如 JLabel, JTextField 或 JButton 组件的文本属性的一种方法。

标识符 一个由字母、数字和下划线所组成的字符序列，用做应用程序单元（比如类或 GUI 组件）的命名。

继承 创建一个基于某个现有类的新类（也称为扩展一个类）。

Integer.parseInt 方法 可返回一个同字符串参数等价的整数。

关键字 由 Java 保留使用的单词。这些单词不能作为标识符来使用。

左花括号({) 用于指示某代码块的开始。

左操作数 某个二目运算符左侧出现的一个表达式。

方法 由一些用于完成某项任务的语句所组成的应用程序片断。方法通常会在执行完任务后返回一些信息。

多行语句 为增强可读性而跨越了多个代码行的一条语句。

乘法运算符 用于对两个数值操作数执行相乘运算并返回作为积的结果的一个星号(*)。

换行 当按下 Enter 键时插入到代码中的一个字符。

操作数 一个同某个运算符（还可能其他的表达式）相结合以完成某项任务的表达式。

保留字 由 Java 保留的且不可作为标识符来使用的单词。参见“关键字”。

从方法中返回一个值 某些方法，当调用时，会向应用程序中调用该方法的语句返回一个值。这个返回的值能够在该语句中进行使用。

右花括号(}) 用于指示某代码块的结束。

右操作数 某个二目运算符右侧出现的一个表达式。

分号(;) 应用程序中用于终止每条语句的字符。

setText 方法 一种用于设置某个组件，如 JLabel, JTextField 或 JButton 文本属性的方法。

语句 可执行某项操作并以一个分号作为结束的代码单元。

String.valueOf 方法 一种将数值转换成文本的方法。

空白 一种属于制表符、空格或者是换行的字符。

Java 类库索引

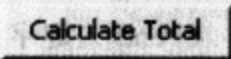
Integer Integer 类含有一些处理整数值的方法。

● 方法

parseInt 返回一个同字符串参数等价的整数。

JButton 该组件允许用户通过对应用程序图形用户界面中的按钮执行点击操作来产生一个事件。

● 运行



● 事件

actionPerformed 可在用户点击 JButton 时发生。

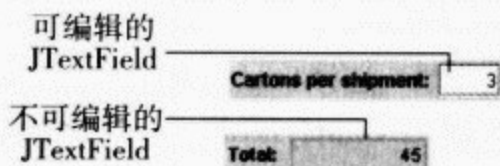
● 方法

setBounds 设置 JButton 的位置与大小。

setText 设置 JButton 的文本属性。

JTextField 该组件允许用户输入信息，同时也用做向用户显示结果。

● 运行



● 方法

getText 返回 JTextField 中所显示的文本。

setBounds 设置 JTextField 的位置与大小。

setEditable 指明用户是否能够编辑 JTextField。若值为 true (默认)，则意味着 JTextField 是一个可编辑的输入 JTextField，若为 false，则意味着 JTextField 是一个不可编辑的输出 JTextField。

setHorizontalAlignment 指定 JTextField 中文本的对齐方式(可选取的值: JTextField.LEFT, JTextField.CENTER, JTextField.RIGHT)。

setText 在 JTextField 中指定显示的文本。

String String 类用于存储和管理文本数据。

● 方法

valueOf 返回与参数等价的字符串。

习题

选择题

4.1 _____ 可表示一个用户的操作，如点击一个 JButton。

- a) 语句 b) 事件 c) 应用程序 d) 函数

4.2 _____ 字符是一个乘法运算符。

- a) 星号 (*) b) 正斜杠 (/) c) 分号 (;) d) 以上答案均不对

4.3 一个 _____ 运算符拥有两个操作数。

- a) 注释 b) 文本 c) 二目 d) 以上答案均不对

4.4 注释 _____。

- a) 有助于增强应用程序的可读性 b) 位于两个正斜杠之后
c) 会被编译程序忽略 d) 以上答案均正确

4.5 Java 语句在遇到 _____ 时将结束。

- a) 正斜杠 (/) 字符 b) 分号 (;) 字符 c) 两个正斜杠 (//) 字符 d) 星号 (*)

4.6 方法 _____ 用于修改文本属性。

- a) changeText b) getText c) setText d) modifyText

- 4.7 用于完成某个特定任务并且可能会返回一个值的代码部分被认为是_____。
- a) 变量 b) 方法 c) 操作数 d) 标识符
- 4.8 Java 关键字是_____。
- a) 标识符 b) 由 Java 保留使用的 c) 区分大小写的 d) (b)和(c)
- 4.9 类的声明是以_____作为结束。
- a) 右花括号 () b) 分号 (;) c) end 关键字 d) class 关键字
- 4.10 方法_____可以将文本转换成数值。
- a) Integer.getInt b) String.valueOf c) Integer.parseInt d) String.value

练习题

- 4.11 (库存清单应用程序的改进版)改进原库存清单应用程序,使之包含一个可允许用户向其输入一周之内所收到货物批次数目的JTextField。假设每批货物都拥有相同数目的箱子(每箱含有相同数目的货物),修改代码以便此库存清单应用程序在计算时使用到这个货物的批次数目。
- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial04\Exercises\Inventory2Enhancement 目录复制到 C:\SimplyJava 目录中。
- b) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\Inventory2Enhancement 进入到当前工作目录中。
- c) 编译模板应用程序 通过键入 javac Inventory.java 对该应用程序进行编译。
- d) 运行模板应用程序 通过键入 java Inventory 运行这个改进后的库存清单模板应用程序。库存清单模板应用程序的 GUI 将会出现在图 4.12 中。注意它与图 4.11 之间的不同。

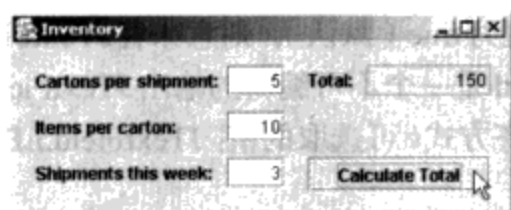


图 4.11 改进后的库存清单应用程序

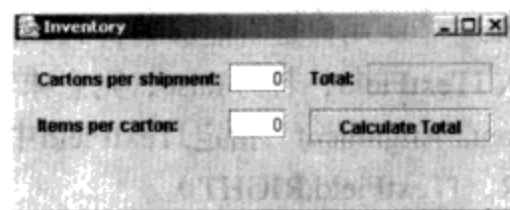


图 4.12 改进后的库存清单模板应用程序

- e) 打开模板文件 在自己的文本编辑器中打开 Inventory.java 文件。
- f) 自定义 Shipments this week:JLabel 在第 70 行的下面,插入一条语句,将 shipmentsJLabel 的 text 属性设置为"Shipments this week:".接着下一行,插入一条语句设置 shipmentsJLabel 的 bounds 属性,使其 y 坐标为 80, x 坐标、宽度和高度与 itemsJLabel 的 x 坐标、宽度和高度相同(参见模板代码第 59 行)。
- g) 自定义 Shipments this week:JTextField 在第 76 行的下面,插入一条语句,将 shipmentsJTextField 的 text 属性设置为"0"。接着下一行,插入一条语句设置 shipmentsJTextField 的 bounds 属性,使其 y 坐标为 80, x 坐标、宽度和高度与 itemsJTextField 的 x 坐标、宽度和高度相同(参见模板代码第 65 行)。接着下一行,插入一条语句设置 shipmentsJTextField 的 horizontalAlignment 属性,使其中的文本靠右对齐(JTextField.RIGHT)。
- h) 改变 Calculate Total JButton 的位置 修改第 99 行里的语句,设置 calculateJButton 的 bounds 属性,使 JButton 的顶部与 shipmentsJTextField 的顶部对齐。
- i) 修改事件处理代码 修改 calculateJButtonActionPerformed 的事件处理程序(第 122 行至第 131 行),使第 126 行至第 129 行中的语句能够将每周收到的货物批次数目同每批货物中箱子的数目与每箱货物的数目的乘积相乘。
- j) 保存应用程序 保存修改后的源代码文件。
- k) 编译完成后的应用程序 通过键入 javac Inventory.java 对应用程序进行编译。
- l) 运行完成后的应用程序 若应用程序能正确编译,通过输入 java Inventory 运行它。将完成后的库存清单应用程序的 GUI 同图 4.11 中所示的 GUI 做一比较,以确信正确地自定义了 GUI 组件和事件处理程序。

- 4.12 (计数器应用程序) 创建一个计数器应用程序。该计数器应用程序中的GUI由一个JTextField和一个JButton组成。JTextField在初始情况下显示0, 但每次用户点击JButton时, JTextField中的值将加1。
- a) 将模板复制到工作目录中 将C:\Examples\Tutorial04\Exercises\Counter目录复制到C:\SimplyJava目录中。
 - b) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\Counter 进入到当前的工作目录中。
 - c) 编译模板应用程序 通过键入 javac Counter.java 对该应用程序进行编译。
 - d) 运行模板应用程序 通过键入 java Counter 运行这个计数器模板应用程序。计数器模板应用程序的GUI将会出现在图 4.14 中。注意它与图 4.13 之间的不同。

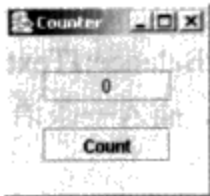


图 4.13 计数器应用程序

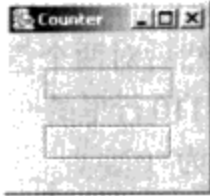


图 4.14 计数器模板应用程序

- e) 打开模板文件 在自己的文本编辑器中打开 Counter.java 文件。
 - f) 自定义 countJTextField 在第 30 行的下面, 插入一条语句将 countJTextField 的 text 属性设置为 "0"。紧接着下一行, 插入一条语句设置 countJTextField 的 horizontalAlignment 属性, 以使 JTextField 中的文本得到居中 (JTextField.CENTER)。
 - g) 自定义 countJButton 在第 38 行的下面, 插入一条语句将 countJButton 的 text 属性设置为 "Count"。
 - h) 在事件处理程序中插入代码 在事件处理程序 countJButtonActionPerformed(第 63 行至第 67 行) 语句体内的第 66 行处插入一条语句, 每当用户点击 Count JButton 时对 countJTextField 中的值加 1, 然后使用这个新得到的值设置 countJTextField 的 text 属性。具体的做法是, 利用加法运算符 (+) 对两个操作数执行相加运算: 通过表达式 1+Integer.parseInt((countJTextField.getText())) 计算出需要进行显示的这一新值。这个表达式的结果还必须利用 String.valueOf 转换成文本才能够在 countJTextField 中显示。
 - i) 保存应用程序 保存修改后的源代码文件。
 - j) 编译完成后的应用程序 通过键入 javac Counter.java 对应用程序进行编译。
 - k) 运行完成后的应用程序 若应用程序能够正确编译, 通过键入 java Counter 运行它。将完成后的计数器应用程序的 GUI 同图 4.13 中所显示的 GUI 做一比较, 以确信正确地自定义了 GUI 组件和事件处理程序。
- 4.13 (银行账户信息应用程序) 创建一个应用程序, 允许用户输入存款的数目 (参见图 4.15)。每当用户点击 Enter JButton 时, 应用程序会将用户在 Deposit amount:JTextField 中输入的存款额添加到当前正在 Balance:JTextField 中显示的余额上, 然后再到 Balance:JTextField 中重新显示出新得到的结果 (注意: 该应用程序以 Sue Purple 作为默认的客户名, 而 12345 将作为一个默认的账户号)。

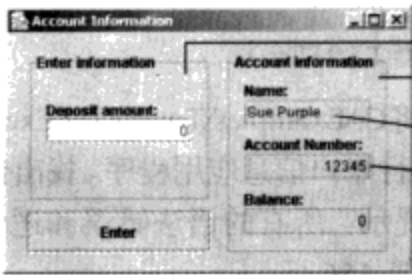


图 4.15 账户信息应用程序

- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial04\Exercises\AccountInformation 目录复制到 C:\SimplyJava 目录中。

- b) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\AccountInformation` 进入到当前的工作目录中。
- c) 编译模板应用程序 通过键入 `javac AccountInformation.java` 对该应用程序进行编译。
- d) 运行模板应用程序 通过键入 `java AccountInformation` 运行这个银行账户信息模板应用程序。在 `Deposit amount:JTextField` 中键入 100。当按下 `Enter JButton` 按钮时, 注意存款额并未添加到余额中, 这是因为读者还没有向 `Enter JButton` 的事件处理程序中添加任何的代码。
- e) 打开模板文件 在自己的文本编辑器中打开 `AccountInformation.java` 文件。
- f) 在事件处理程序中插入代码 在文本编辑器中, 将屏幕滚动至事件处理程序 `enterJButtonActionPerformed` 所在的位置处 (第 145 行至第 149 行)。在该事件处理程序语句体内第 148 行, 插入一条语句取得 `Deposit amount:JTextField(depositJTextField)` 和 `Balance:JTextField(balanceJTextField)` 中的文本数值, 然后对两个值相加并把结果显示在 `Balance:JTextField` 中。可采用类似图 4.10 中的第 108 行至第 110 行里的一些技术。接着下一行, 插入一条语句将 `Deposit amount:JTextField` 的 `text` 属性设置为 "0"。
- g) 保存应用程序 保存修改后的源代码文件。
- h) 编译完成后的应用程序 通过键入 `javac AccountInformation.java` 对应用程序进行编译。
- i) 运行完成后的应用程序 若应用程序能正确编译, 通过键入 `java AccountInformation` 运行它。
- j) 测试应用程序 在 `Deposit amount:JTextField` 中键入 100, 然后按下 `Enter JButton`。 `Balance:JTextField` 中现在应显示出 100 且 `Deposit amount:JTextField` 中应显示为 0。在 `Deposit amount:JTextField` 中输入 50, 然后按下 `Enter JButton` 按钮。 `Balance:JTextField` 中现在应显示为 150, 而 `Deposit amount:JTextField` 中现在应显示为 0。

4.14 (说出这段代码的作用) 当 `priceJTextField` 中键入 2, `hammersJTextField` 中键入 14 以后, 点击一个名为 `calculateJButton` 的 `JButton`, 便能够计算出特定数量的榔头的总价钱。假设有下列代码, 则在点击以后, 其结果是什么?

```
1 private void calculateJButtonActionPerformed( ActionEvent event )
2 {
3     totalPriceJTextField.setText( String.valueOf(
4         Integer.parseInt( priceJTextField.getText() ) *
5         Integer.parseInt( hammersJTextField.getText() ) ) );
6
7 } // calculateJButtonActionPerformed
```

4.15 (找出代码中的错误) 下面的这个事件处理程序是在用户点击一个 `Multiply JButton` 时便开始执行的。假设每一个 `JTextField` 中都含有这样的一个表示某整数值的文本。找出该段代码中的错误 (一个或一个以上)。

```
1 private void multiplyJButtonActionPerformed( ActionEvent event )
2 {
3     resultJTextField.setText( leftOperandJTextField.getText() *
4         rightOperandJTextField.getText() );
5
6 } // multiplyJButtonActionPerformed
```

4.16 (银行账户信息应用程序的调试练习) 将 `C:\Examples\Tutorial04\Exercises\DebuggingExercise` 目录复制到 `C:\SimplyJava` 目录下。编译并运行银行账户信息应用程序。找出任何出现的语法错误和逻辑错误, 最终应使该应用程序能够正确运行 [提示: 所有的语法错误和逻辑错误均出自事件处理程序中的代码 (位于该应用程序中第 146 行至第 156 行)]。

挑战题

4.17 (改进银行账户信息应用程序) 修改习题 4.13, 使用户既能输入取款额又能输入存款额 (参见图 4.16)。当点击 `Enter JButton` 按钮时, 其余额也会得到正确地更新。

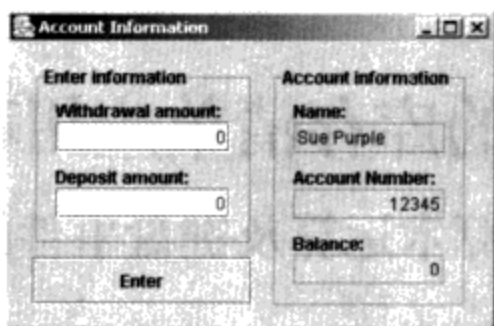


图 4.16 改进后的银行账户信息应用程序

- a) 将模板复制到工作目录中 将C:\Examples\Tutorial04\Exercises\AccountInformationEnhancement目录复制到C:\SimplyJava目录中。
- b) 打开命令提示符窗口改变目录 选择Start → Programs → Accessories → Command Prompt打开一个命令提示符窗口。键入cd C:\SimplyJava\AccountInformationEnhancement进入到当前工作目录中。
- c) 编译模板应用程序 通过键入javac AccountInformation.java对该应用程序进行编译。
- d) 运行模板应用程序 通过键入java AccountInformation运行这个改进后的银行账户信息模板应用程序。在Withdrawal amount:JTextField中键入50，在Deposit amount:JTextField中输入100。当按下Enter JButton时，注意存款额并未改变，这是因为还没有往Enter JButton的事件处理程序中添加任何代码。
- e) 打开模板文件 在自己的文本编辑器中打开AccountInformation.java文件。
- f) 在事件处理程序中插入代码 在文本编辑器中，将屏幕滚动至事件处理程序enterJButtonActionPerformed所在的位置处（第163行至第167行）。在该事件处理程序语句体内第166行，插入一条语句，分别取得Withdrawal amount:JTextField(withdrawalJTextField)，Deposit amount:JTextField(depositJTextField)和Balance:JTextField(balanceJTextField)中的文本数值，然后执行计算： $balance + deposit\ amount - withdrawal\ amount$ ，并把结果显示在Balance:JTextField中（可以采用类似图4.10中第108行至第110行里的一些技术）。最后，在该事件处理程序中，插入两条语句将Withdrawal amount:JTextField和Deposit amount:JTextField中的text属性均设置为“0”。
- g) 保存应用程序 保存修改后的源代码文件。
- h) 编译完成后的应用程序 通过键入javac AccountInformation.java对应用程序进行编译。
- i) 运行完成后的应用程序 若应用程序能够正确编译，通过键入java AccountInformation运行它。
- j) 测试应用程序 在Withdrawal amount:JTextField中键入50，在Deposit amount:JTextField中键入100，然后按下Enter JButton。Balance:JTextField中现在应显示为50，而且Withdrawal amount:和Deposit amount:JTextField中现在应显示为0。再为Deposit amount:JTextField中键入50，然后按下Enter JButton按钮。Balance:JTextField中现在应显示出为100，而Deposit amount:JTextField中现在应显示为0。

教程 5 改进的库存清单应用程序

引入变量、内存、算术运算及键盘事件的概念



教学目标

在本教程中，读者将学到以下内容：

- 声明变量
- 针对 JTextField 处理 keyPressed 事件
- 通过使用变量学习与内存相关的基本概念
- 使用算术运算符优先级规则
- 为应用程序的调试设置断点
- 使用调试程序中的 run, stop, cont 和 print 命令

在前一个教程中，经开发后得到的库存清单应用程序，会将接收到的货箱数目同每箱中的货物数目执行相乘运算并计算出该公司收到的全部货物的总数量。通过使用 JTextField 完成从键盘中读入用户输入以及输出计算后的结果。同时，利用往 JFrame 中添加的一个 JButton，通过编程实现了用户在点击该 JButton 时所进行的响应操作。在本教程中，读者将使用另外一些编程概念来继续改进这个库存清单应用程序，这些概念包括：变量、键盘事件以及算术运算，等等。

5.1 探试改进后的库存清单应用程序

在本教程中，读者将通过使用变量改进前一个教程中的库存清单应用程序。读者将学到内存的概念，这有助于理解应用程序是如何在计算机中进行执行的。回顾教程 4 中的库存清单应用程序，它以用户提供的信息作为计算所收货物数量的基础，即箱子的数目与每箱中教科书的数目。经改进后的这个应用程序需满足下面的需求：

应用程序需求分析

库存清单管理员注意到这个库存清单应用程序存在一个小小的失误。尽管应用程序能够计算出正确的结果，但是这个结果在新的数据输入之后仍然继续显示。而此输出只有当库存清单管理员再次按下 Calculate JButton 时，才会得到改变。程序设计人员应对这个库存清单应用程序进行改进，使得用户向任意一个 JTextField 中输入新的信息时，原结果就将被删除掉，以避免用户对计算结果的准确性产生疑虑。

我们将以这个完成后的应用程序的探试作为开始。然后，学习另外的一些 Java 技术并最终创建一个属于自己的应用程序。



探试改进后的库存清单应用程序

1. 定位到完成后的应用程序 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\Examples\Tutorial05\CompletedApplication\Inventory3` 将目录改变到这个完成后的库存清单应用程序的目录下面。

2. 运行库存清单应用程序 在命令提示符窗口下键入 `java Inventory` 运行该应用程序 (如图 5.1 所示)。
3. 计算一批货物的数量 在 `Cartons per shipment:JTextField` 中键入 5, 在 `Items per carton:JTextField` 中键入 6。点击 `Calculate Total JButton`。结果将显示在作为输出的 `Total:JTextField` 中 (如图 5.2 所示)。
4. 输入新值 当用户在任意一个 `JTextField` 中输入某个新值时, 在 `Total:JTextField` 中显示的结果将被删除。比如, 输入 13 作为箱子数目的一个新值。注意当输入进 1 时, 最近一次计算的结果立刻被清除掉 (如图 5.3 所示)。用于完成这项任务的 Java 技术将在本教程的后面得到解释。

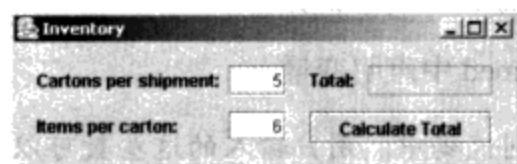


图 5.1 运行完成后的库存清单应用程序

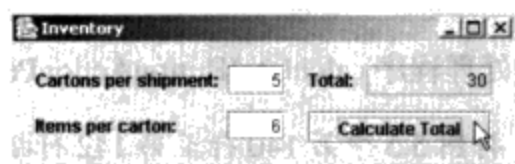


图 5.2 计算一批货物的数量

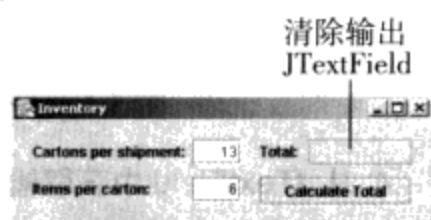


图 5.3 改进后的库存清单应用 (程序在新的值输入以后)

对作为输出的 `JTextField` 立刻执行了清除操作 (用以避免可能引起的困惑)。

5. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
6. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

5.2 变量



好的编程习惯

通常情况下, 变量名是以一个小写字母作为开始的。变量名中除第一个单词外, 每一个单词都应该以一个大大的字母作为开始 (例如, `firstNumber`)。

变量是用来保存数据的存储单元, 非常类似 `JLabel` 中作为存储向用户显示文本的 `text` 属性。与 `JLabel` 的 `text` 属性所不同的是, 变量允许开发人员对数据进行存储和管理, 而不必将数据显示给用户。变量同样也允许开发人员在不使用 GUI 组件的条件下存储数据。变量可以存储如数字、日期、时间以及其他一些类似的数据。然而, Java 中所使用的每一个变量都确切地对应着一种信息类型。例如, 一个存储数字的变量并不能够用来存储作为某种名称 (或者是其他的文本) 的值。

Java 中, 所有变量必须被声明以后才能够在应用程序中使用。变量的声明包括变量的类型 (表示变量中用以存储数据的数据类型) 以及一个可选用的初始化值 (表示变量中所存储数据的初始值)。在本教程中, 将学习用做声明整数变量的 `int` 类型, 即变量的值必须是一个整数。



好的编程习惯

在变量名中只使用字母和数字。

接下来, 将学习如何通过变量进行编程。变量名必须是一个有效的标识符, 所谓标识符, 正如在教程 3 中了解到的, 是由字母、数字、下划线 (`_`) 以及美元符号 (`$`) 所组成的一个字符序列, 但标识符决不能是一个关键字。另外, 标识符不能以数字作为开始, 也不能含有空格。

在库存清单应用程序中使用变量

1. 将模板复制到工作目录中 将 `C:\Examples\Tutorial05\TemplateApplication\Inventory3` 目录复制到 `C:\SimplyJava` 目录中。
2. 打开库存清单应用程序的模板文件 在文本编辑器中打开模板文件 `Inventory.java`。
3. 向事件处理程序 `calculateJButtonActionPerformed` 中添加变量的声明 将图 5.4 中的第 135 行至第 138 行添加到 `calculateJButtonActionPerformed` 事件处理程序中。第 136 行至第 138 行表示变量的声明, 变

量的声明是以每种变量的类型作为开始的（本例中，所使用的类型为 int）。关键字 int 表示被声明的变量只能存储整数型的数值（例如，919，0，-11）。而单词 cartons，items 和 result 属于变量的名称。

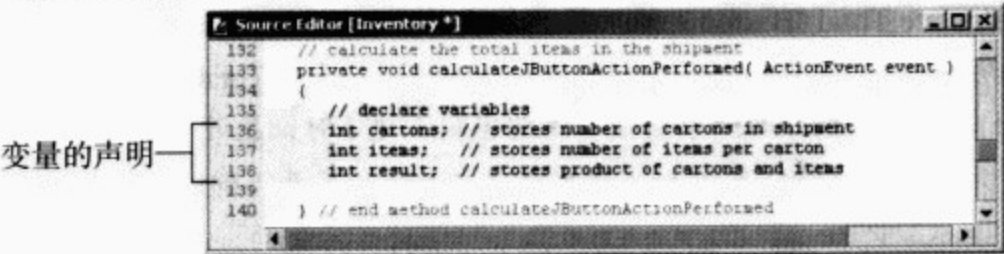


图 5.4 在事件处理程序 calculateJButtonActionPerformed 中声明变量

- 4. 从 JTextField 中获取输入 将图 5.5 中第 140 行至第 142 行插入到代码中。用户输入的这些数可以通过 JTextField 的 text 属性来获取。而这些数据必须经过方法 Integer.parseInt 转换成整数值，然后再将这些所得到的整数利用赋值运算符(=)，分别赋值给变量 cartons（参见第 141 行）和 items（参见第 142 行）。赋值运算符会将其右侧表达式的值复制到其左侧的变量中。第 141 行实际按照某个特定的顺序完成了三项操作。首先，通过调用 getText 方法取得 cartonsJTextField 的 text 属性。接着，Integer.parseInt 将一个字符串值转换成一个 int 值。最后，该整数被放置在变量 cartons 中。第 141 行可以读做“cartons 被赋予了一个由 Integer.parseInt(cartonsJTextField.getText()) 所返回的整数值”。第 142 行是将 itemsJTextField 的 text 属性以整数值的形式赋予了变量 items。
- 5. 保存应用程序 保存修改后的源代码文件。
- 6. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\Inventory3 进入到当前的工作目录中。
- 7. 编译应用程序 通过键入命令 javac Inventory.java，对该应用程序进行编译。
- 8. 运行应用程序 若此应用程序能够正确编译，可以通过键入 java Inventory 来运行它。图 5.6 显示了更新后的应用程序的运行结果。分别在两个 JTextField 中输入 5 和 6，按下 Calculate Total JButton。这时，Total:JTextField 中并未显示出计算的结果，因为还没有给应用程序添加相应的代码；读者将在随后的示例中添加这些代码。

将用户的输入
赋值给变量
JTextField 中
输入的数值

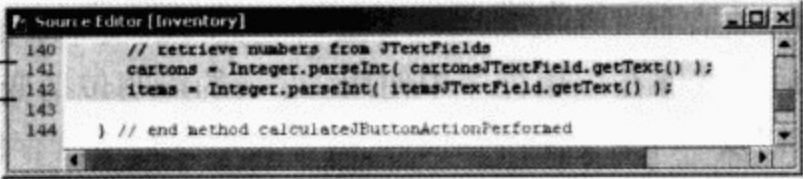


图 5.5 通过事件处理程序 calculateJButtonActionPerformed 取得

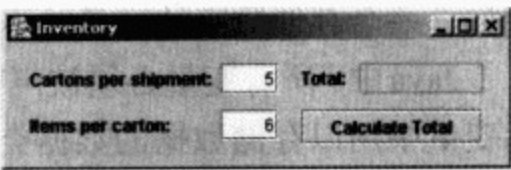


图 5.6 更新后的库存清单应用程序

- 9. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- 10. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

刚才所看到的类型为 int 的变量都属于整数型的变量。而类型为 double 的变量可以存储拥有一个小数点的数。这些数被称为浮点数，它们能够存储一些如 2.3456，0.0 和 -845.4680 这样的数值。类型为 double 的变量能够容纳比类型为 int 的变量更大（或者更小）的值。这些在 Java 中已得到定义的类型，如 int，被认为是基本数据类型。所有基本类型的类型名同样也属于关键字。一共有 8 种基本类型，参见图 5.7 中的列表。回顾曾经提到过，关键字是由 Java 保留使用的，因此关键字不能作为标识符来使用（有关 Java 关键字的完整列表请参考附录 E）。

基本类型			
boolean	byte	char	short
int	long	float	double

图 5.7 Java 的基本类型

第 141 行和第 142 行（如图 5.5 所示）是将用户在 JTextField 中输入的两个值转换成 int，然后再把它们赋值给 cartons 和 items 变量。现在，我们就将利用这些变量来计算所收到的教科书的总数量。

在计算中使用变量

- 1. 完成乘法运算 在自己的文本编辑器中，对前一示例中最后所添加的那条语句的末尾处增加一个空白行（第 142 行）。然后，插入图 5.8 中的第 144 行至第 145 行。第 145 行这条语句是将 int 型变量 cartons 的值与 int 型变量 items 的值相乘，并通过赋值运算符（=）将结果赋值给变量 result。该条语句可以读做“result 被赋予了 cartons * items 的值”（绝大多数的计算是通过赋值语句来完成的）。当有变量出现在计算中时，该变量所存储的当前值会应用到该计算之中。注意，计算过程中并不会修改 cartons 或者 items 中的值。

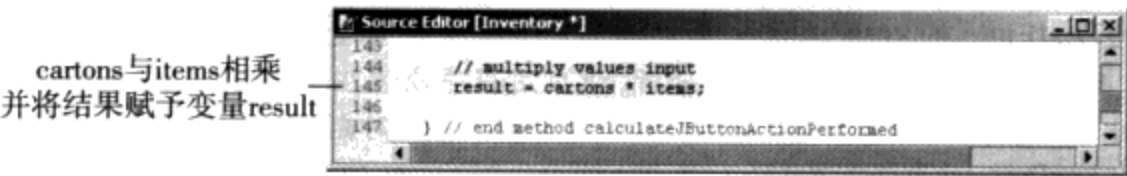


图 5.8 在一条赋值语句中完成两个变量的乘法运算

- 2. 显示结果 将图 5.9 中第 147 行至第 148 行添加到 calculateJButtonActionPerformed 事件处理程序中。在计算完成以后，第 148 行通过设置属于输出的 JTextField，即 totalResultJTextField 中的 text 属性完成乘法运算后的结果显示。回顾教程 4 中曾经提到过，String.valueOf 能够将一个数转换成一个用于在 JTextField 中进行显示的 String。

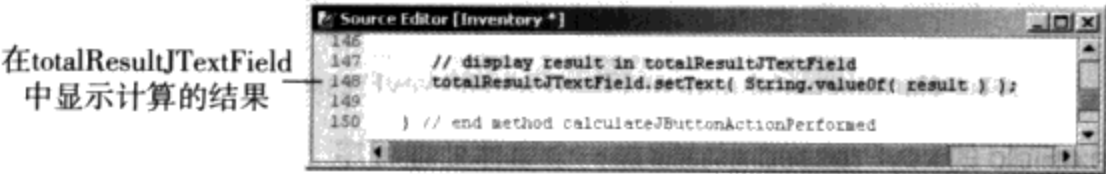


图 5.9 显示计算的结果

- 3. 保存应用程序 保存修改后的源代码文件。
- 4. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\Inventory3 进入到当前的工作目录中。
- 5. 编译应用程序 通过键入命令 javac Inventory.java，对该应用程序进行编译。
- 6. 运行应用程序 若此应用程序能够正确编译，可以通过键入 java Inventory 来运行它。图 5.10 显示了更新后的应用程序的运行结果。在 Cartons per shipment:JTextField 中输入 5，在 Items per cartons:JTextField 中输入 6。当按下 Calculate Total JButton 时，Total:JTextField 中将显示出结果（30）。

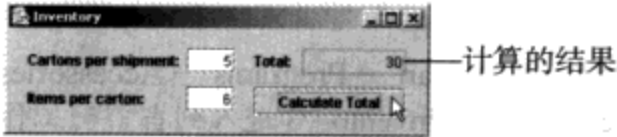


图 5.10 显示乘法运算的结果

- 7. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- 8. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

自测题

- 1. 变量名必须是一个 ____。
a) 关键字 b) 有效的标识符 c) 以上两个答案均正确 d) 以上答案均不对
- 2. 一些已在 Java 中定义的类型，如 int，被认为是 ____ 类型。
a) 创建 b) 现存 c) 已定义的 d) 基本

答案：1) b 2) d

5.3 针对 JTextField 处理 keyPressed 事件

读者或许已经注意到，本教程一开始的应用程序需求分析中所提到的一个小小失误（或称 bug），仍正处于应用程序之中。也就是指，尽管 totalResultJTextField 能够显示出当前的结果，但是，若在任意的一个或者是两个用做输入的 JTextField 中输入一个新的数时，原结果理应失效。而到目前为止，这个作为显示的结果只有在点击 Calculate Total JButton 时才会再次变得有效，而这种做法，很可能会对使用应用程序的用户产生困惑。



处理 keyPressed 事件

1. 为 cartonsJTextField 的 keyPressed 事件添加一个事件处理程序 将图 5.11 中第 155 行插入到代码中。按照本教程中有关应用程序的需求分析，该应用程序应该在每次用户改变任意一个输入 JTextField 中的文本时，就清除掉 totalResultJTextField 中的显示。任何时刻，若是在 JTextField 中按下某个键，便可以激发出 keyPressed 事件。第 155 行使用 setText 清除 totalResultJTextField 中的值。在第 155 行中使用的这个标记 ""（两个紧挨着的双引号）称为空字符串，其字符串值将不包含任何字符。利用空字符串替换掉在 totalResultJTextField 的 text 属性中所存储的内容，因而也就清除掉了屏幕上显示的结果。

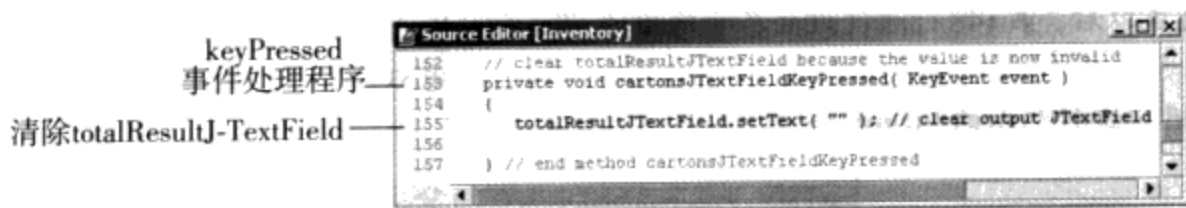


图 5.11 Cartons per shipment:JTextField 的 keyPressed 事件处理程序

2. 为 itemsJTextField 的 keyPressed 事件添加一个事件处理程序 既然不论哪一个输入 JTextField 首先将被改变，其结果都应该被清除掉，那么，还应当处理 itemsJTextField 的 keyPressed 事件。将图 5.12 中的第 162 行插入到这个新的事件处理程序中。注意该行与图 5.11 中的第 155 行一样，可以完成相同的任务。这是因为都希望发生一个相同的操作，即对 JTextField 的清除。

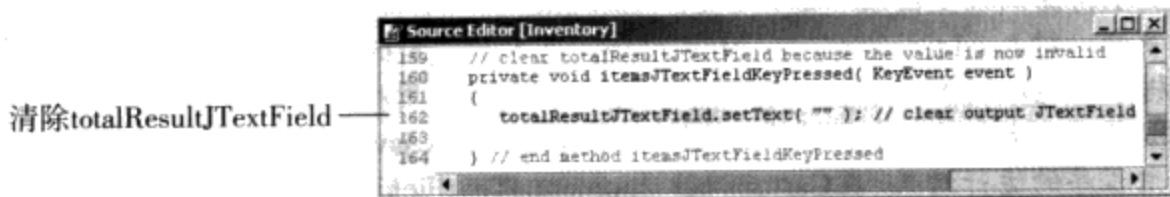


图 5.12 Items per carton:JTextField 的 keyPressed 事件处理程序

3. 保存应用程序 保存修改后的源代码文件。
4. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\Inventory3 进入到当前的工作目录中。
5. 编译应用程序 通过键入命令 javac Inventory.java，对该应用程序进行编译。
6. 运行应用程序 若此应用程序能够正确编译，可以通过键入 java Inventory 来运行它。图 5.13 显示了更新后的应用程序的运行结果。为了测试应用程序，在 Cartons per shipment:JTextField 中输入 8，在 Items per cartons:JTextField 中输入 7。当点击 Calculate Total JButton 时，输出 JTextField 中将显示数字 56。接下来，在 Items per carton:JTextField 中输入 9 以确信 keyPressed 事件处理程序可以清除掉输出 JTextField 中所显示的内容(如图 5.13 所示)。点击 Calculate Total 计算新的结果 (72)，然后改变 Cartons per shipment:JTextField 中的值以确信能够同样清除输出 JTextField 中的结果。

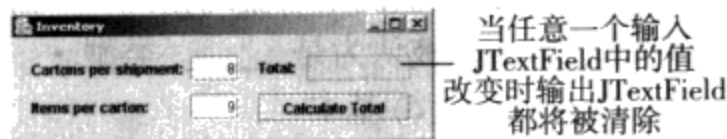


图 5.13 测试完成后的库存清单应用程序

7. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。

8. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

图 5.14 中给出了改进后的库存清单应用程序的完整源代码。本教程中，凡需要添加、查看或是修改的代码均在图中相应的代码行中做了突出显示。

```
1 // Tutorial 5: Inventory.java
2 // Calculates the number of items in a shipment based on the number
3 // of cartons received and the number of items per carton.
4 import java.awt.*;
5 import java.awt.event.*;
6 import javax.swing.*;
7
8 public class Inventory extends JFrame
9 {
10     // JLabel and JTextField for cartons per shipment
11     private JLabel cartonsJLabel;
12     private JTextField cartonsJTextField;
13
14     // JLabel and JTextField for items per carton
15     private JLabel itemsJLabel;
16     private JTextField itemsJTextField;
17
18     // JLabel and JTextField for total items per shipment
19     private JLabel totalJLabel;
20     private JTextField totalResultJTextField;
21
22     // JButton to initiate calculation of total items per shipment
23     private JButton calculateJButton;
24
25     // no-argument constructor
26     public Inventory()
27     {
28         createUserInterface();
29     }
30
31     // create and position GUI components; register event handlers
32     public void createUserInterface()
33     {
34         // get content pane for attaching GUI components
35         Container contentPane = getContentPane();
36
37         // enable explicit positioning of GUI components
38         contentPane.setLayout( null );
39
40         // set up cartonsJLabel
41         cartonsJLabel = new JLabel();
42         cartonsJLabel.setText( "Cartons per shipment:" );
43         cartonsJLabel.setBounds( 16 , 16 , 130, 21 );
44         contentPane.add( cartonsJLabel );
45
46         // set up cartonsJTextField
47         cartonsJTextField = new JTextField();
48         cartonsJTextField.setText( "0" );
49         cartonsJTextField.setBounds( 148, 16 , 40 , 21 );
50         cartonsJTextField.setHorizontalAlignment( JTextField.RIGHT );
51         contentPane.add( cartonsJTextField );
52         cartonsJTextField.addKeyListener(
53
54             new KeyAdapter() // anonymous inner class
55             {
56                 // method called when user types in cartonsJTextField
```



```

57         public void keyPressed( KeyEvent event )
58         {
59             cartonsJTextFieldKeyPressed( event );
60         }
61     } // end anonymous inner class
62
63     ); // end call to addKeyListener
64
65     // set up itemsJLabel
66     itemsJLabel = new JLabel();
67     itemsJLabel.setText( "Items per carton:" );
68     itemsJLabel.setBounds( 16 , 48, 104 , 21 );
69     contentPane.add( itemsJLabel );
70
71     // set up itemsJTextField
72     itemsJTextField = new JTextField();
73     itemsJTextField.setText( "0" );
74     itemsJTextField.setBounds( 148, 48, 40 , 21 );
75     itemsJTextField.setHorizontalAlignment( JTextField.RIGHT );
76     contentPane.add( itemsJTextField );
77     itemsJTextField.addKeyListener(
78         new KeyAdapter() // anonymous inner class
79         {
80             // method called when user types in itemsJTextField
81             public void keyPressed( KeyEvent event )
82             {
83                 itemsJTextFieldKeyPressed( event );
84             }
85         } // end anonymous inner class
86     ); // end call to addKeyListener
87
88     // set up totalJLabel
89     totalJLabel = new JLabel();
90     totalJLabel.setText( "Total:" );
91     totalJLabel.setBounds( 204 , 16 , 40 , 21 );
92     contentPane.add( totalJLabel );
93
94     // set up totalResultJTextField
95     totalResultJTextField = new JTextField();
96     totalResultJTextField.setBounds( 244, 16 , 86 , 21 );
97     totalResultJTextField.setHorizontalAlignment(
98         JTextField.RIGHT );
99     totalResultJTextField.setEditable( false ); // output only
100     contentPane.add( totalResultJTextField );
101
102     // set up calculateJButton
103     calculateJButton = new JButton();
104     calculateJButton.setText( "Calculate Total" );
105     calculateJButton.setBounds( 204 , 48, 126, 24 );
106     contentPane.add( calculateJButton );
107     calculateJButton.addActionListener(
108         new ActionListener() // anonymous inner class
109         {
110             // method called when calculateJButton is pressed
111             public void actionPerformed( ActionEvent event )
112             {
113                 calculateJButtonActionPerformed( event );
114             }
115         }
116     );
117
118
119
120

```

```

121         } // end anonymous inner class
122
123     ); // end call to addActionListener
124
125     // set properties of application's window
126     setTitle( "Inventory" ); // set title bar text
127     setSize( 354, 112 ); // set window size
128     setVisible( true ); // display window
129
130 } // end method createUserInterface
131
132 // calculate the total items in the shipment
133 private void calculateJButtonActionPerformed((ActionEvent event) )
134 {
135     // declare variables
136     int cartons; // stores number of cartons in shipment
137     int items;   // stores number of items per carton
138     int result;  // stores product of cartons and items
139
140     // retrieve numbers from JTextFields
141     cartons = Integer.parseInt( cartonsJTextField.getText() );
142     items = Integer.parseInt( itemsJTextField.getText() );
143
144     // multiply values input
145     result = cartons * items;
146
147     // display result in totalResultJTextField
148     totalResultJTextField.setText( String.valueOf( result ) );
149
150 } // end method calculateJButtonActionPerformed
151
152 // clear totalResultJTextField because the value is now invalid
153 private void cartonsJTextFieldKeyPressed( KeyEvent event )
154 {
155     totalResultJTextField.setText( "" ); // clear output JTextField
156
157 } // end method cartonsJTextFieldKeyPressed
158
159 // clear totalResultJTextField because the value is now invalid
160 private void itemsJTextFieldKeyPressed( KeyEvent event )
161 {
162     totalResultJTextField.setText( "" ); // clear output JTextField
163
164 } // end method itemsJTextFieldKeyPressed
165
166 // main method
167 public static void main( String[] args )
168 {
169     Inventory application = new Inventory();
170     application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
171
172 } // end method main
173
174 } // end class Inventory

```

图 5.14 库存清单应用程序的完整代码

自测题

1. 在 Java 中, _____ 可由 "" 来表示。

- a) 空字符 b) 空字符串 c) 空值 d) 以上答案均不对

2. 使用 _____ 方法可以清除 JTextField 中所显示的任何文本。
- a) clearText b) removeText c) resetText d) setText

答案: 1) b 2) d

5.4 内存的概念

变量名，如 cartons, items 以及 result，对应计算机内存中实际存储的位置。每一个变量都拥有名称、类型、长度以及相应的值。在图 5.14 中的库存清单应用程序的代码中，有一条赋值语句：

```
cartons = Integer.parseInt( cartonsJTextField.getText() );
```

当其执行时，cartonsJTextField 中所存储的用户输入被转换成了一个类型为 int 的值。这一 int 值将被放置在专为 cartons 而分配的某个内存单元处。整个过程是，假设用户向 Cartons per shipment: JTextField 中输入 12，该输入将被存储在 cartonsJTextField 中（在 text 属性中）。当点击 Calculate Total 时，此 JButton 的事件处理程序便开始执行，它利用方法 Integer.parseInt 将用户输入转换成了一个 int 值并将该 int 值放入 cartons 所在的内存单元处，如图 5.15 所示。

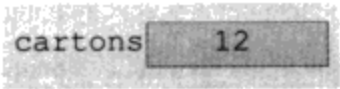


图 5.15 展示变量 cartons 的名称与值的内存单元

一旦有值被放入某个内存单元中，此值将替换掉该位置中先前所存入的值。也就是说，前一个值可能会覆盖掉（或者丢失）。因此，针对某个内存单元的编写过程被认为是具有破坏性质的。

图 5.7 中的每种基本类型都拥有各自的长度，即内存中用于存储该类型值的字节数目。例如，一个 int 类型的数会占据内存中的 4 个字节，所表示值的范围是在 -2 147 483 648 至 +2 147 483 647 之间。对于某些应用程序，这个范围是有些小了。如果是这样的话，还可以考虑使用类型 long，它以内存中 8 个字节的长度进行存储，所能够表示的整数范围是从 -9 223 372 036 854 775 808 到 +9 223 372 036 854 775 807 之间。附录 F 总结了全部的基本类型，包括它所占据的字节长度以及该类型变量所存储值的范围。

假设用户在 Items per carton:JTextField 中输入 10 并点击 Calculate Total，这会使该 JButton 的事件处理程序开始执行。图 5.14 中第 142 行：

```
items = Integer.parseInt( itemsJTextField.getText() );
```

是将存储在 itemsJTextField 中的字符串转换成一个 int 并将此 int 值放入 items 所在的内存单元处。此时的内存如图 5.16 所示。

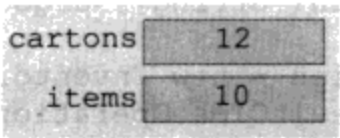


图 5.16 向 cartons 和 items 中赋值以后的内存单元

当点击 Calculate Total JButton 时，其事件处理程序将开始执行，当执行到第 145 行时，会完成一个乘法运算并将所得到的积放入 result 变量中。语句：

```
result = cartons * items;
```

用于执行这一乘法运算并替换（即改写）result 中先前所存放的值。当 result 被计算出来以后，此时的内存如图 5.17 所示。注意，这时 cartons 和 items 中的值完全和计算 result 之前时是一样的。尽管

这些值要在计算机完成计算时进行使用，但它们并不会得到改写。这表明，从内存中的某个位置处读取一个值的过程是非破坏性的（这意味着该值并不会改变）。

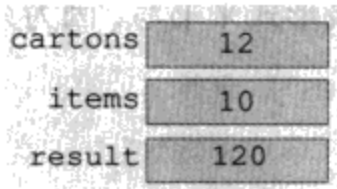


图 5.17 完成乘法运算后的内存单元

自测题

- 1. 当一个变量被放置在内存中的某个单元中时，此值会 _____ 该单元中先前所存放的值。
a) 复制 b) 替换 c) 将其自身添加到 d) 移动
- 2. 当某个值从内存中读出时，此值会 _____。
a) 被改写 b) 用一个新的值进行替换 c) 移动至内存中一个新的位置处 d) 不发生改变

答案：1) b 2) d

5.5 算术运算

大多数应用程序都需要完成一些算术运算。在上一个教程中，通过使用乘法运算符 (*), 便可以执行一个算术乘法运算。在图 5.18 中，对一些算术运算符的使用过程进行了总结。需要注意的是，一些特定的符号却并不允许在代数中使用。比如，用星号 (*) 表示乘法运算，百分号 (%) 表示的是求余运算符（随后将进行解释），正斜杠 (/) 则表示除法运算。

图 5.18 中所有的算术运算符都是二目运算符，即每一个运算符都需要有两个操作数。例如，在表达式 `sum + value` 中包含了一个二目运算符 + 以及两个操作数 `sum` 和 `value`。Java 同时也提供了所谓的单目运算符，这些运算符只需要有一个操作数。比如，利用单目运算符加 (+) 和单目运算符减 (-)，程序员便可以编写出一些使用如，+9（一个正数）和 -19（一个负数）等完成某些表达式的运算（注意：单目加是很少被使用的，因为默认情况下数都是正数）。

Java运算符	算术运算符	代数表达式	Java表达式
加	+	$f + 7$	<code>f + 7</code>
减	-	$p - c$	<code>p - c</code>
乘	*	bm	<code>b * m</code>
除	/	x / y or $\frac{x}{y}$ or $x \div y$	<code>x / y</code>
求余	%	$r \text{ mod } s$	<code>r % s</code>

图 5.18 算术运算符

整数之间的除法运算需要有两个整数 (int) 作为操作数，其最终结果将得到一个作为整数值的商。比如，表达式 `7/4` 的结果为 1，表达式 `17/5` 的结果为 3。注意到整数除法运算结果中，任何属于小数的部分均被抛弃掉了（称为截尾）——并不存在四舍五入的问题。但除法运算符若是使用在带有浮点数（包含小数点的数）的除法运算中时，其结果将是一个浮点数。比如，表达式 `7.0/4.0` 的结果为 1.75，表达式 `17.0/5.0` 的结果为 3.4。

求余运算符 (%) 能够得到除法运算以后的余数。例如，表达式 `x%y` 能够得到 x 除以 y 以后所得的余数。因而，`7%4` 将得到 3，`17%5` 将得到 2。通常，这一运算符更多的是使用在 int 型的操作数中，但它也可以应用在其他类型中。应用求余运算符常常可以解决一些有趣的问题，比如，查找

某数是否是另一个数的倍数。例如，若 a 和 b 都是数，如果 a 是 b 的倍数，则 $a \% b$ 的结果为 0。 $8 \% 3$ 的结果为 2，所以 8 不是 3 的倍数。但是 $8 \% 2$ 和 $8 \% 4$ 的结果都是 0，所以 8 既是 2 的倍数，也是 4 的倍数。

Java 中的算术表达式必须按照直线形式来书写，因为只有这样才能将它们输入进计算机中。例如，7.1 除以 4.3 不可以写为：

$$\frac{7.1}{4.3}$$

而只能是按照直线形式书写成 $7.1/4.3$ 。

在 Java 表达式中，括号的使用方式同代数表达式中的使用方式是一致的。例如，为使 a 乘以 $b + c$ 的和，应写为：

$$a * (b + c)$$

在算术表达式中应用这些运算符，Java 是按照一个预先定义好的运算符优先级规则的顺序来执行的，这通常与代数中的顺序是一致的。这些规则能够保障 Java 按照一个正确的顺序使用这些运算符。图 5.19 遵照一定的顺序列出了这些规则。

运算符优先级规则

1. 首先是使用单目加 (+) 与单目减 (-)：若一个表达式中含有多个单目加与单目减运算符，则这些运算符将按照从右至左的方向运行

2. 接下来是使用乘法 (*)、除法 (/) 及求余 (%) 运算：若一个表达式中包含多个乘法、除法和求余运算，则这些运算符将按照从左至右的方向运行

3. 最后是采用加法 (+) 和减法 (-) 运算：若一个表达式中包含多个加法和减法运算，则这些运算符将按照从左至右的方向运行

图 5.19 运算符优先级规则

下面，让我们按照运算符的优先级规则来考察一些表达式。每一个例子中，均会列出一个代数表达式和对应的一个 Java 表达式。

首先是计算三个数的平均值：

代数： $m = \frac{(a + b + c)}{3}$

Java: $m = (a + b + c) / 3$

同代数中的一样，括号用于为计算中的一些表达式进行分组。前一个表达式中的括号是必需的，因为除法要比加法拥有更高的优先级。也就是说，是将整个 $(a + b + c)$ 的结果除以 3。如果括号被省略，即，得到 $a + b + c / 3$ ，则将计算：

$$a + b + \frac{c}{3}$$

显然这会得到一个错误的结果（属于逻辑错误）。同时还需要注意的是，变量 m 的赋值操作是最后才完成的，也就是，在加法运算完成之后才进行的。

接下来是一个直线形等式：

代数： $y = mx + b$

Java: $y = m * x + b$

其中，并不需要使用括号。乘法运算应首先执行，因为乘法运算比加法运算拥有更高的优先级。而赋值是最后发生的，因为它比乘法和加法的优先级都低。注意，如果是：

$$y = mx + b$$

则很可能是一个逻辑错误，因为 mx 在 Java 中被认为是某个变量的一个有效变量名—— mx 并不像在代数中意味着“ m 乘以 x ”。

为了更好地理解运算符优先级规则，最后考察这样的一个表达式 $y = ax^2 + bx + c$ ，是如何进行计算的：



该条语句下面带有圆圈的数字代表的是 Java 使用这些运算符的顺序。在 Java 中， x^2 需表示成 $x * x$ ，因为没有所谓的乘幂运算符。同样，注意赋值运算符是最后才被使用到的，因为它比其他任何算术运算符的优先级都低。

同代数中一样，在表达式中放置一些不必要的括号是可以接受的，为的是增强表达式的易读性——这些括号被称为冗余括号。例如，在上一个赋值语句中可以使用一些冗余括号，以突出相应的各项：

$$y = (a * x * x) + (b * x) + c$$



好的编程习惯

在复杂的算术表达式中使用冗余括号有助于阅读表达式。

自测题

1. 在 Java 中，在编写算术表达式的时候，必须 _____ 才可以输入进计算机中。
a) 使用括号 b) 在多行上 c) 以直线形式 d) 以上答案均不对
2. 赋值运算符 (=) 右侧的表达式总是在赋值发生之 _____ 进行计算。
a) 前 b) 后 c) 同一时刻 d) 以上答案均不对

答案：1) c 2) a

5.6 调试程序:断点设置与 run, stop, cont 和 print 命令

在 2.4 节中，读者曾了解到存在着两种类型的错误——语法错误与逻辑错误，同时也学习了如何去除代码中的语法错误。

逻辑错误并不能阻止应用程序的成功编译，但它确实会在应用程序运行时产生出错误的结果。Java SDK 中含有一个名为调试程序 (debugger) 的软件，通过它可以帮助开发人员监测应用程序的执行，以便查找并去除相应的逻辑错误。

调试程序是最重要的应用程序开发工具之一。我们将把学习使用断点作为整个调试程序学习的开始。所谓断点是指一个能够在任意可执行的代码行中进行设置的标记。当应用程序执行到一个断点时，执行被暂停，同时，允许开发人员考察一些变量的值来确信是否存在着逻辑错误。比方说，通过考察某个存储计算结果的变量值来确信该计算是否能得以正确执行。注意，假若是对某个不可执行的代码行（比如一个注释）设置了一个断点的话，调试程序将显示出一条错误的信息。我们将通过使用一些断点及各种调试程序命令来考察事件处理程序 `calculateJButtonActionPerformed` 中所声明的一些变量值。



使用调试程序:断点设置与 run, stop, cont, chear 和 print 命令

1. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\Inventory3` 进入到当前的工作目录中。

- 2. 编译库存清单应用程序 Java 的调试程序只能使用那些在编译时选用了 -g 选项的.class 文件，这是因为该选项生成了一些需由编译程序进行使用的编译信息，从而帮助开发人员对应用程序做出调试。通过输入带有 -g 命令行选项的命令 javac -g Inventory.java，重新编译这个库存清单应用程序。
- 3. 启动调试程序 在命令提示符窗口下键入 jdb（如图 5.20 所示）。该命令将启动 Java 的调试程序，之后便可以使用调试程序中的一些有用的性质了。



图 5.20 启动 Java 的调试程序

- 4. 在调试程序中运行应用程序 通过键入 run Inventory（如图 5.21 所示）使库存清单应用程序通过调试程序来运行。假若在依赖调试程序运行应用程序之前未设置任何的断点，则应用程序的运行结果与使用 java 命令所运行的结果是一致的。
- 5. 关闭正在运行的应用程序并重新启动调试程序 为了能够正确使用调试程序，通常在运行应用程序之前至少应设置一个断点。点击正在运行的应用程序的关闭按钮将其终止，然后通过键入 jdb（如图 5.22 所示）重新启动调试程序。
- 6. 在 Java 中插入断点 以下将为应用程序中的一个特定的代码行设置一个断点。所使用到的代码行的行号将与图 5.14 中源代码中的行号保持一致。通过键入 stop at Inventory:145 为源代码中的第 145 行设置一个断点（如图 5.23 所示）。stop 命令用于在该命令之后所指定的某一行处插入一个断点。读者可根据需要设置多个断点。比如，通过键入 stop at Inventory:148 为代码中第 148 行再设置另外的一个断点（如图 5.23 所示）。当应用程序运行时，其执行会暂停在含有断点的代码行处。而当调试程序暂停应用程序的执行时，应用程序将被认为是处于中止模式。



图 5.21 通过调试程序运行库存清单应用程序



图 5.22 重新启动调试程序

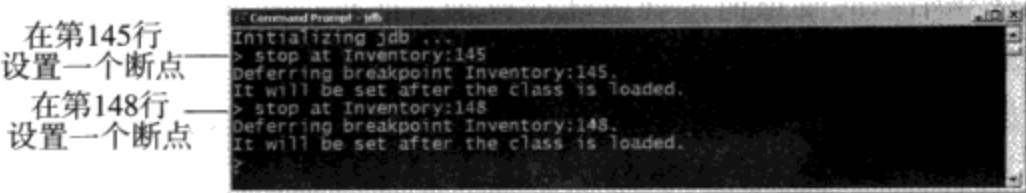


图 5.23 设置两个断点

- 7. 运行应用程序 键入 run Inventory 运行应用程序，之后调试过程便开始了（如图 5.24 所示）。注意命令提示符窗口中的最后两行，分别指出了在第 145 行和第 148 行设置的这两个断点。调试程序称每一个断点为一个“延时断点”，因为只有在每一个断点设置好以后，应用程序才又开始在调试程序中继续运行。

重新启动库存清单应用程序

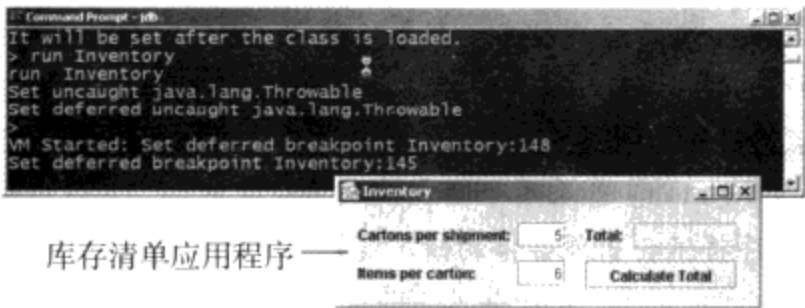


图 5.24 重新启动库存清单应用程序

- 8. 开始调试过程 在两个 JTextField 中分别键入 7 和 10 并点击 Calculate Total，使之继续运行。应用程序在执行到第 145 行的断点时会暂停下来（如图 5.25 所示）。观察命令提示符窗口中由调试程序所显示的信息。调试程序会告知读者到达了一个断点，然后便显示出该行（参见第 145 行）所在处的源代码。而此行是即将执行的下一条语句。
- 9. 使用 cont 命令恢复执行 键入 cont。应用程序将继续运行直至到达下一个断点时为止（参见第 148 行），而那时候，命令提示符窗口会给出一个通告（如图 5.26 所示）。
- 10. 考察某个变量的值 键入 print cartons。当前在 cartons 变量中存储的值将得到显示（如图 5.27 所示）。print 命令允许开发人员窥探计算机中某个变量的值。该命令能够帮助开发人员查找并消除代码中的逻辑错误。使用 print 命令还可以输出在变量 cartons，items 以及 result 中所存储的值（如图 5.27 所示）。

断点到达
下一个即将执行的代码行

暂停库存清单应用程序



图 5.25 到达第一个断点

到达第二个断点



图 5.26 执行到达第二个断点

在 cartons 中的值
在 items 中的值
在 result 中的值

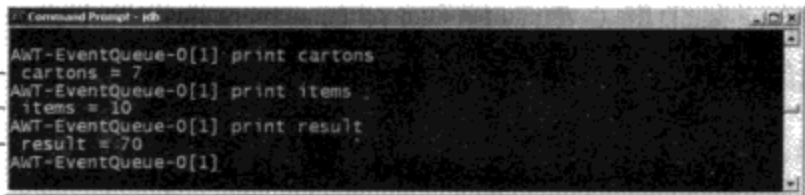


图 5.27 考察三个变量中的值

- 11. 继续执行应用程序 键入 cont 继续执行该应用程序。如果没有其他的断点，应用程序会继续执行并将结果显示在 Total:JTextField 中（如图 5.28 所示）。此时，该应用程序将不再处于终止模式。

恢复执行

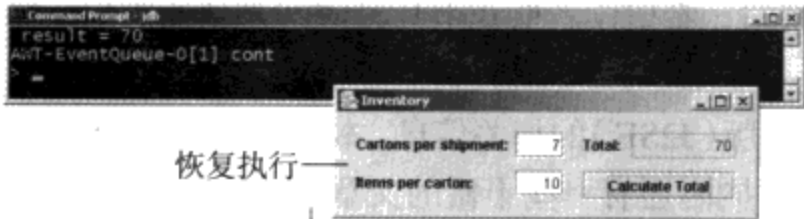


图 5.28 继续执行应用程序

- 12. 删除断点 键入 clear 命令，通过一个列表显示出应用程序中的所有断点（如图 5.29 所示）。若是键入 clear Inventory:145，则可以去除掉该应用程序中的第一个断点。利用同样的方法再去掉第二个断点。

现在,再键入clear查看应用程序中剩余的断点。此时,调试程序应指示出不存在有任何的断点设置了(如图5.29所示)。

13. 执行不含断点的应用程序 在运行着的库存清单应用程序的两个JTextField中分别输入4和9,然后按下Calculate Total JButton。如果两个断点被成功删除的话,新的输出结果(36)将显示在输出JTextField中,而应用程序将不会出现暂停的情况(如图5.30所示)。

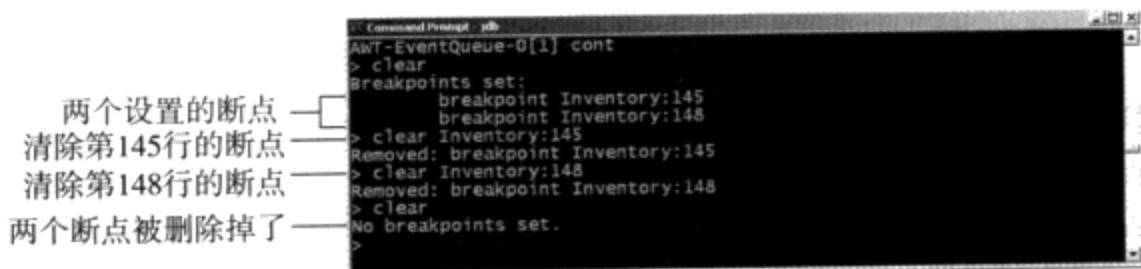


图 5.29 删除断点

14. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。当应用程序结束时调试程序也将停止执行(如图5.31所示)。

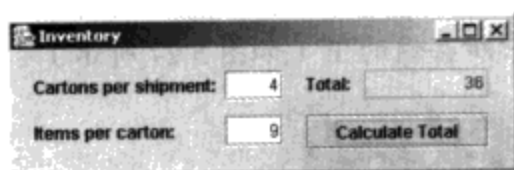


图 5.30 执行不含断点设置的应用程序



图 5.31 退出调试程序

15. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

在本节中,读者学习了如何使用调试程序并设置相应的断点,通过print命令考察了一个正在运行的应用程序中的一些变量。同时读者还学习了如何使用cont命令继续执行在某个断点处暂停执行的应用程序,以及如何使用clear命令来去除断点。

自测题

- 断点不能设置在_____中。
 - 注释
 - 可执行的代码行
 - 赋值语句
 - 算术语句
- 当应用程序在某个断点处暂停执行时,下一条即将执行的语句将位于该断点所在行语句_____的语句。
 - 之前
 - 之后
 - 处
 - 以上答案均不对

答案: 1) a 2) c

5.7 Internet 与 Web 资源

这里推荐一些 Web 网站,希望读者能花上片刻时间来访问它们。

java.sun.com/j2se

该网站上有Java 2平台标准版(J2SE)的相关信息。本书采用的是J2SE 1.4.1版。该站点就有关J2SE的所有特征做了一个概述,同时还有一些不错的文章、新闻和相关技术的链接。

www.aewnet.com/java

该网站上含有一些相关的训练、目录、文章以及访问其他Java站点的链接。

www.java.ittoolbox.com

该网站提供了一些关于Java的新闻、技术讨论、代码和教程。

java.sun.com/search

该站点上提供了一个专为回答开发人员所遇到的 Java 问题的搜索引擎。

www.ibiblio.org/javafaq

该站点提供了一些 Java 新闻、使用 API 时常会遇到的问题及其他的一些 Java 资源。

www.developer.com/java

该站点上含有许多 Java 的新闻、教程和资源。

java.sun.com/products/jpda/doc/soljdb.html

该站点对 Java 的调试程序进行了描述并给出了如何使用它的一些额外信息。

5.8 小结

到现在为止,读者已在库存清单应用程序中添加了一些变量。通过使用这些变量,可以得出同教程 4 中的库存清单应用程序同样的结果。利用 `keyPressed` 事件,对这个库存清单应用程序进行了改进,使之能够在用户改变任意一个输入 `JTextField` 时,便可执行清除输出 `JTextField` 中内容的代码。

读者学习到了有关内存的概念,其中包括如何读取变量和改写变量。我们将把这些概念应用到以后教程中所构建的应用程序之中,而这些应用程序也将主要依靠变量来发挥作用。接着,还学习了如何在 Java 中执行算术运算并研究了用于保证算术表达式正确执行的运算符优先级规则。最后,学习了如何在调试程序中插入和删除断点。利用断点,可以允许开发人员暂停应用程序的执行,这样便能通过使用调试程序中的 `print` 命令考察出变量的值。通过这种能力能够帮助开发人员查找并改正应用程序中的逻辑错误。

在下一个教程中,将设计一个工资额计算器的图形用户界面并编写该应用程序的一些代码。我们还将学习伪代码的概念,它是一种能够帮助开发人员设计应用程序的语言。还将通过使用调试程序中的 `print` 命令对某些 Java 表达式进行计算,随后再利用调试程序中的 `set` 命令改变应用程序中一些变量的值。

技术小结

声明变量

- 指定变量的类型,例如,将一个变量指定为 `int` 类型。
- 使用一个有效的标识符作为变量的名字。

处理 `JTextField` 的 `keyPressed` 事件

- 将代码插入到 `JTextField` 的事件处理程序中,该事件处理程序会在用户向 `JTextField` 中输入数据时开始执行。

从内存单元中读取一个值

- 在赋值语句的右侧或者是在一个表达式中使用某个变量的名字。

替换内存单元中的值

- 利用变量名,后跟赋值运算符 (`=`),随后是一个可得出某个新值的表达式来完成。

表示正数或负数

- 使用单目加 (`+`) 和单目减 (`-`) 表示。

执行算术运算

- Java 中的算术表达式需按照直线形式进行书写。

- 利用运算符优先级规则判定需选用的运算符的次序。
- 使用运算符“+”执行加法运算。
- 使用运算符“-”执行减法运算。
- 使用运算符“*”执行乘法运算。
- 使用运算符“/”执行除法运算。
- 使用运算符“%”计算除法运算后所得到的余数。
- 使用运算符“=”将某个计算结果赋值给一个变量。
- 使用括号“()”对执行计算的表达式进行分组。

运行调试程序

- 在命令提示符窗口中，键入jdb启动调试程序。

依赖调试程序运行应用程序

- 当调试程序启动以后，通过键入run命令并指定一个要运行的应用程序的类名。

设置一个断点

- 当调试程序运行时，使用stop命令并指定一个类名，随后跟一个冒号和需设置断点的行号。

在进入中止模式后恢复应用程序的执行

- 当Java调试程序运行时，使用cont命令恢复程序的执行。

考察断点状况下某个变量的值

- 当执行到某个断点时，利用print命令并在随后跟上一个希望查看其中内容的某个变量的名字。

查看应用程序中所有断点的一个列表

- 当调试程序运行时，使用clear命令可以列出所有的断点。

删除断点

- 当调试程序运行时，使用clear命令并指定一个类的名字，随后跟一个冒号和一个行号，便可删除掉相应行上的这个断点。

关键术语

算术运算符 一些如+，-，*，/和%等可用于执行计算的运算符。

赋值运算符 赋值运算符(=)是将其右侧表达式的值复制到其左侧的变量中。

星号(*) 表示乘法运算的算术运算符。

中止模式 调试程序使应用程序的执行停止在某个断点时的一种应用程序的状态。

断点 一个在调试程序中可对任意可执行代码行进行设置的标记，该标记会使应用程序在到达指定的代码行时暂停执行。设置断点的一个原因是，希望应用程序执行到该断点时，可以考察某些变量的值。

bug 可阻止应用程序正确执行的错误。

clear 调试程序命令 用于删除某个断点的命令。

cont 调试程序命令 调试过程中，一个在断点到达以后用于恢复程序执行的命令。

调试程序 允许开发人员对应用程序执行监测以便查找并删除逻辑错误的软件。

声明变量 指定应用程序中所使用的某个变量的类型和名字。

声明 用于指定某变量的名字和类型的代码。

破坏性质 在对内存单元进行编写的过程中，会使该单元中以前所存储的值得到改写或丢失。

double 一种用于存储浮点数的类型。

空字符串("") 一个不包含任何字符的字符串。

正斜杠 (/) 一个表示除法运算的算术运算符。

浮点数 拥有一个小数点的数，如 2.3456，0.0，-845.4680。

-g 编译程序选项 用于生成编译程序所使用的编译信息，以便帮助开发人员调试应用程序的一种选项。

初始值 某变量开始时的值。

整数 数的一种，如 919，-11，0。

int 类型 一种用于存储整数值的数据类型。

jdb 一个在命令提示符窗口中输入以启动 Java 调试程序的命令。

keyPressed 事件 当 JTextField 中有任何按键按下时，一种可产生的事件。

逻辑错误 一种并不能阻止应用程序成功编译，但会使应用程序在运行时产生错误结果的错误。

变量名 应用程序中用于访问或修改某个变量值的标识符。

非破坏性质 内存单元的一个读取过程，该过程并不会修改内存单元中的值。

%(求余运算符) 一种在除法运算后可得到余数的运算符。

基本类型 Java 中预先定义好的一种数据类型。基本数据类型包括 boolean，byte，char，short，int，long，float 和 double。

print 调试程序命令 当应用程序在执行过程中停止于调试过程中的某个断点处时，一个用于显示某变量值的命令。

冗余括号 在执行计算的时候，作为清晰执行过程的额外括号。这些并不影响计算结果的括号，是完全可以删除掉的。

运算符优先级规则 用于判定表达式中运算符使用次序的规则。

run 调试程序命令 一种在 Java 调试程序中用于开始某个应用程序执行的命令。

变量的长度 用于存储某类型变量值的字节数。例如，int 是以内存中 4 个字节长度来存储的，而 double 则是以 8 个字节长度来进行存储的。

stop 调试程序命令 用于在某个特定可执行代码行中设置断点的命令。

直线形式 算术表达式必须遵守的一种书写形式，通过这种形式便能完成相关 Java 代码的输入。

整数除法中的截尾 整数除法运算结果中的任何小数部分将被抛弃。

变量的类型 指定某变量中可存储的数据种类及所存储值的取值范围。例如，一个 int 变量能够存储的整数范围是从 -2 147 483 648 到 +2 147 483 647 之间。

单目运算符 一种只拥有一个操作数的运算符（如，单目 + 和单目 -）。

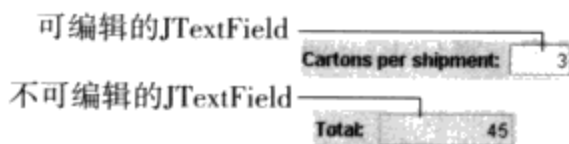
变量的值 在变量所在内存单元中存储的数据。

变量 可表示计算机内存中的一个位置，用于存储应用程序所使用的值。

Java 类库索引

JTextField 该组件允许用户输入信息，同时也可作为向用户显示结果来使用。

● 运行



● 事件

keyPressed 当任何按键作用在 JTextField 中时将产生此事件。

● 方法

getText 返回 JTextField 中所显示的文本。

setBounds 设置 JTextField 的位置与大小。

setEditable 指明用户是否能够编辑 JTextField。若值为 true（默认），则意味着 JTextField 是一个可编辑的输入 JTextField，若为 false，则意味着 JTextField 是一个不可编辑的输出 JTextField。

- l) 显示结果 使用JTextArea的方法append将抵押年限长度及月度支付额显示在outputJTextArea中。需使用两个tab字符以确保月度支付额能够正确地出现在第二列上(因为年限长度位于表头的第一列上)。
- m) 在while语句中自增counter 切记每一次的循环过程都应为counter做自增的运算。
- n) 保存应用程序 保存修改后的源代码文件。
- o) 打开命令提示符窗口改变目录 选择Start→Programs→Accessories→Command Prompt 打开一个命令提示符窗口。键入cd C:\SimplyJava\MortgageCalculator2进入到当前的工作目录中。
- p) 编译应用程序 通过键入javac MortgageCalculator.java, 编译该应用程序。
- q) 运行完成后的应用程序 若此应用程序能正确编译, 通过键入java MortgageCalculator来运行它。使用图8.24中的值对这一应用程序的运行进行测试。

8.13 (办公用品应用程序) 创建一个应用程序, 允许某用户制作一张需要购买的办公用品列表, 如图8.25所示。用户将在一个JTextField中输入用品名称, 然后点击Buy JButton, 该用品则被添加到了JTextArea中。而Clear JButton会将所有用品从JTextArea中删除。

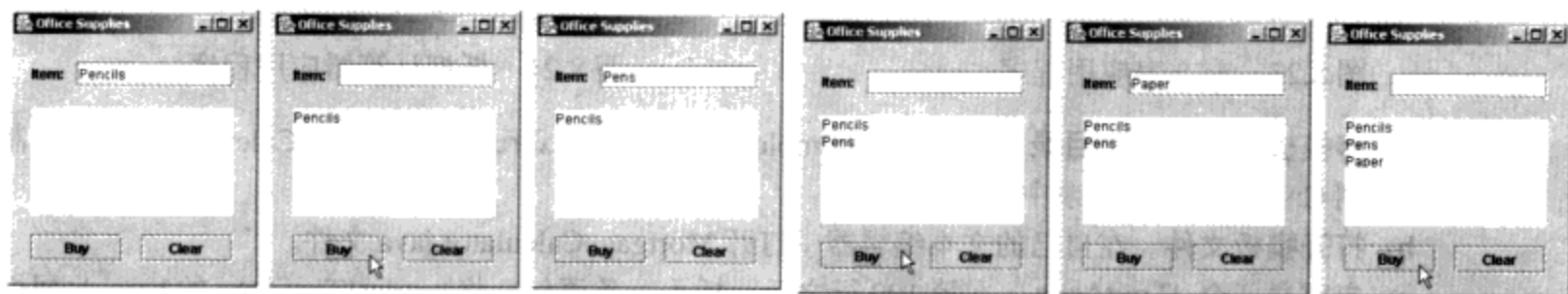


图 8.25 办公用品应用程序

- a) 将模板复制到工作目录中 将C:\Examples\Tutorial08\Exercises\OfficeSupplies目录复制到C:\SimplyJava目录中。
- b) 打开模板文件 在自己的文本编辑器中打开OfficeSupplies.java文件。
- c) 为Buy JButton事件处理程序添加代码 向buyJButtonActionPerformed事件处理程序(起始于第105行)添加一条语句, 从itemJTextField中取得用户输入, 并将该用户输入附加到outputJTextArea中。再添加另外一条语句, 清除itemJTextField中的显示。
- d) 为Clear JButton事件处理程序添加代码 向clearJButtonActionPerformed事件处理程序(紧接着buyJButtonActionPerformed事件处理程序)添加一条语句, 使用JTextArea的方法setText清除outputJTextArea中显示的内容。
- e) 保存应用程序 保存修改后的源代码文件。
- f) 打开命令提示符窗口改变目录 选择Start→Programs→Accessories→Command Prompt 打开一个命令提示符窗口。键入cd C:\SimplyJava\OfficeSupplies进入到当前的工作目录中。
- g) 编译应用程序 通过键入javac OfficeSupplies.java, 编译该应用程序。
- h) 运行完成后的应用程序 若此应用程序能正确编译, 通过键入java OfficeSupplies来运行它。利用图8.25中的值对这一应用程序进行测试。

8.14 (说出这段代码的作用) 以下代码, 其执行的结果是什么?

```

1 int x = 1 ;
2 int mysteryValue = 1 ;
3
4 while ( x < 6 )
5 {
6     mysteryValue *= x;
7     x++;
8 }
9
10 displayJLabel.setText( String.valueOf( mysteryValue ) );

```

8.15 (找出代码中的错误) 寻找下列代码中的错误:

- a) 假设变量 `counter` 已得到声明并被初始化为 1。此循环将对 1~100 的数进行求和。

```
1 while ( counter <= 100 )
2 {
3     total += counter;
4 }
5
6 counter++;
```

- b) 假设变量 `counter` 已得到声明并被初始化为 1000。此循环将从 1000 遍历至 1。

```
1 while ( counter > 0 )
2 {
3     displayJLabel.setText( String.valueOf( counter ) );
4     counter--;
5 }
```

- c) 假设变量 `counter` 已得到声明并被初始化为 1。此循环需执行 5 次并把 1~5 之间的数附加到一个 `JTextArea` 中。

```
1 while ( counter < 5 )
2 {
3     numbersJTextArea.append( String.valueOf( counter ) );
4     counter++;
5 }
```

- 8.16 (使用调试程序)(奇数应用程序) 奇数应用程序将显示出从 1 到用户输入数之间的所有奇数。图 8.26 中显示了这一应用程序的正确输出。在这一练习中, 将利用调试程序寻找并改正这个应用程序中所包含的错误。

- 将模板复制到工作目录中** 将 `C:\Examples\Tutorial08\Exercises\Debugger\OddNumbers` 目录复制到 `C:\SimplyJava` 目录中。
- 打开命令提示符窗口改变目录** 选择 `Start → Programs → Accessories → Command Prompt` 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\OddNumbers` 进入到当前的工作目录中。
- 运行应用程序** 键入 `java OddNumbers`, 运行这个应用程序。在 `Upper limit:JTextField` 中输入一个值, 然后点击 `View JButton`。注意观察 `JButton` 仍处于被按下的状态并且也无任何的输出结果出现。这是因为, 在 `viewJButtonActionPerformed` 事件处理程序中含有一个无限循环语句。
- 关闭正在运行的应用程序** 通过按住键盘上的 `Ctrl` 键和 `C` 键, 然后点击命令提示符窗口便可关闭正在运行的应用程序。
- 因调试需要编译应用程序** 输入带有 `-g` 命令行选项的命令 `javac -g OddNumbers.java` 编译应用程序。
- 启动调试程序** 在命令提示符窗口中, 键入 `jdb`。
- 设置断点** 通过键入 `stop at OddNumbers:94`, 在方法 `viewJButtonActionPerformed` 中第 94 行的位置处设置一个断点。
- 在调试程序中运行应用程序** 在调试程序中通过键入 `run OddNumbers` 来运行这一奇数应用程序。在 `Upper limit:JTextField` 中键入 10 并点击 `View JButton`。应用程序会在执行到第 94 行时进入中止模式。
- 定位逻辑错误** 使用调试程序中的 `print` 命令考察变量 `counter` 和 `limit` 的值。这两个变量的值应该分别为 1 和 10。使用 `cont` 命令从第 94 行的断点处继续执行该应用程序。使得循环的第一轮迭代能够顺利完成, 紧接着, 应用程序将再次因第 94 行而进入到中止模式。使用调试程序中的 `print` 命令考察变量 `counter` 和 `limit` 的值。它们应分别为 2 和 10。可是, 现在 `counter` 的值仍然为 1, 而 `limit` 的值则变成了 11。如果有必要, 可以多次使用 `cont` 和 `print` 来确定代码中出现的问题。
- 打开模板文件并改正代码中的错误** 在自己的文本编辑器中打开 `OddNumbers.java` 文件, 将屏幕滚动至第 92 行至第 101 行所在的位置处, 找到代码中的错误并进行改正。

- k) 保存应用程序 保存修改后的源代码文件。
- l) 编译应用程序 通过键入 `javac OddNumbers.java`, 编译该应用程序。
- m) 运行完成后的应用程序 若此应用程序能够正确编译, 通过键入 `java OddNumbers` 来运行它。
再次向 `Upper limit:JTextField` 中输入一个整数, 然后点击 `View JButton` 来对这一应用程序进行测试。
- n) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- o) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

挑战题

8.17 (To-Do 列表应用程序) 需采用一个可作为 To-Do 列表而使用 `JTextArea`。通过一个 `JTextField`, 输入每一项, 然后点击 `JButton` 将其添加至 `JTextArea` 中。所输入的项目应该显示在一个经过编号的列表中, 如图 8.27 所示。为实现这一编号功能, 应使用 `JTextArea` 中用于返回其行号的方法 `getLineCount`。例如, 以下语句是将 `outputJTextArea` 中所显示的行号赋值给了 `int` 型的变量 `counter`:

```
int counter = outputJTextArea.getLineCount();
```

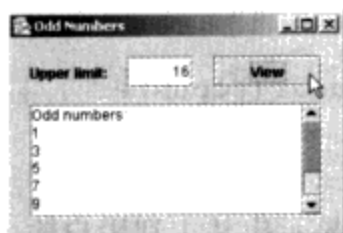


图 8.26 奇数应用程序的正确输出

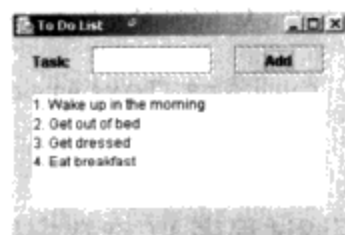


图 8.27 To-Do 列表应用程序

- a) 将模板复制到工作目录中 将 `C:\Examples\Tutorial08\Exercises\ToDoList` 目录复制到 `C:\SimplyJava` 目录中。
- b) 打开模板文件 在自己的文本编辑器中打开 `ToDoList.java` 文件。
- c) 为 `Add JButton` 的事件处理添加代码 将用于获取 `outputJTextArea` 中所显示行号的代码添加至 `addJButtonActionPerformed` 的事件处理程序中 (起始于第 82 行), 从 `taskJTextField` 中取得输入并在前面附上一个行号。当用户的输入被添加到 `outputJTextArea` 上以后, 清除 `taskJTextField` 中的内容。
- d) 保存应用程序 保存修改后的源代码文件。
- e) 打开命令提示符窗口改变目录 选择 `Start → Programs → Accessories → Command Prompt` 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\ToDoList` 进入到当前的工作目录中。
- f) 编译应用程序 通过键入 `javac ToDoList.java`, 编译该应用程序。
- g) 运行完成后的应用程序 若此应用程序能正确编译, 通过键入 `java ToDolist` 来运行它。利用图 8.27 中的值对这一应用程序进行测试。



教程9 班级平均分应用程序

介绍 do...while 循环语句

教学目标

在本教程中，读者将学到以下内容：

- do...while 语句
- 理解计数器控制的循环
- 显示输入对话框
- 启用及禁用 JButton

本教程将继续讨论循环语句。在前一个教程中，通过 while 循环语句了解到，该循环语句是在执行循环体语句之前先来判断循环-继续条件。本教程将介绍另外一种循环语句——do...while 循环语句，该循环语句是在执行完循环体语句之后才进行判断的。因而，其循环体中的语句至少要被执行一次。

我们还将学习如何启用及禁用组件。当某个组件（如 JButton）被禁用时，将不再响应与用户之间的交互。这有助于防止应用程序中的错误。本教程还将介绍输入对话框，它允许应用程序等待用户输入数据并对之进行处理。

9.1 探试班级平均分应用程序

班级平均分应用程序必须满足以下的需求：

应用程序需求分析

某教师需要为学生人数为 10 人的一个班进行测验。测验后的成绩是一些范围在 0~100 之间的整数（包括 0 和 100）。该教师希望能开发一个应用程序，帮助计算该班学生经过测验后的平均分。应用程序还将通过一个输入对话框来输入这些成绩。

班级的平均分等于参加测验学生的成绩总和，除以学生的总人数。利用计算机解决这一问题的算法是：先输入每个学生的成绩，然后，求出成绩总和，计算相应的平均值，最后显示出结果。

我们将以这个完成后的应用程序的探试作为开始。之后，学习另外一些 Java 技术，最终创建一个属于自己的应用程序。



探试班级平均分应用程序

1. 定位到完成后的应用程序 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\Examples\Tutorial09\CompletedApplication\ClassAverage` 将目录改变到完成后的班级平均分应用程序的目录下面。

2. 运行班级平均分应用程序 在命令提示符窗口下键入 java ClassAverage，运行该应用程序（如图 9.1 所示）。Get Grades JButton 会使应用程序显示出一列供用户输入成绩的输入对话框，而 Average JButton 将使该应用程序计算出这些成绩的平均值。输入的所有成绩都将显示在 Grade list:JTextArea 中，而计算出的平均值则显示在 Class average:JTextField 中。可以看到，在应用程序开始执行前，Average JButton 是被禁用的（其文本颜色为灰色），点击它并不能使其调用相应的事件处理程序。这是因为，用户不能在输入成绩前就去计算平均值。JTextArea 应足够大以便能显示出 10 行，并将每一个成绩用一行来显示。同时，利用一个 JLabel 标识出 JTextArea 中将显示的信息类别。



GUI 设计提示

大多数的 JTextArea 都应配有一个描述其性质的 JLabel，用于指明所要显示的输出信息。为这样的 JLabel 使用语句大写形式。

3. 输入测验成绩 点击 Get Grades JButton。这时会显示出一个输入对话框，每次只能输入一个成绩（如图 9.2 所示）。请注意，这里的输入对话框有点类似于消息对话框，只是它还包含了一个允许用户输入数据的 JTextField。在输入对话框中输入 85，作为第一个测验成绩，然后点击 OK JButton。点击 OK JButton 会导致对话框消失（从屏幕中删除）。所输入的值（85），可被应用程序访问到并在平均值计算中进行使用。该应用程序通过一个循环语句来显示多个输入对话框。因此，当输入一个成绩并按下 OK 按钮时，该应用程序会立刻显示出另一个输入对话框并要求用户继续输入下一个成绩。这项工作将一直持续到整个循环的结束。

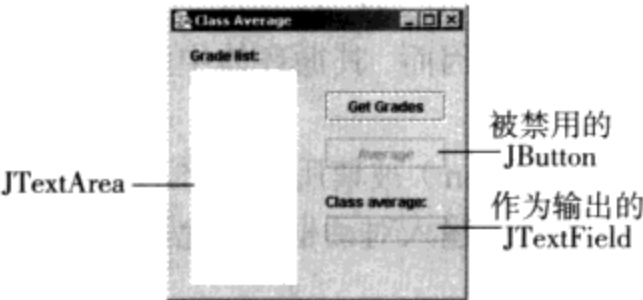


图 9.1 运行完成后的班级平均分应用程序

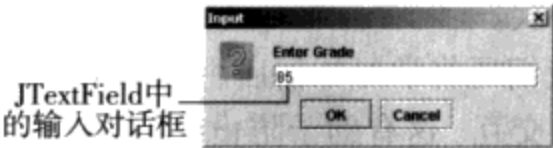


图 9.2 在班级平均分应用程序中输入成绩

4. 将步骤 3 重复 9 次 按照图 9.3 输入另外的 9 个成绩，在每输入一项后，点击 OK JButton。当输入完成以后，所有的 10 个成绩都将显示在 JTextArea 中（如图 9.3 所示）。此时，应用程序会启用 Average JButton，因而，可通过点击它计算并显示出这些成绩的平均值。注意观察 Average JButton 内的文本周围有一个淡淡的蓝色矩形边框，表示该组件已被选取和启用。当某个组件被选取时，我们说它拥有了该应用程序的焦点。这样的一个矩形边框有助于将用户的注意力集中到下一个即将使用的组件上。GUI 设计的一个重要部分便是帮助用户去理解：使用哪些组件才可以同应用程序实现更好的交互。我们将在本教程的后面学习如何设置某个组件的焦点。当某个 JButton 拥有焦点以后，可采用鼠标点击或者是按下空格键的方式来完成其相应的操作。

5. 计算平均值 采用按下空格键的方式来选取 Average JButton。这时，10 个成绩的平均值将显示在 Class average:JTextField 中（如图 9.4 所示）。

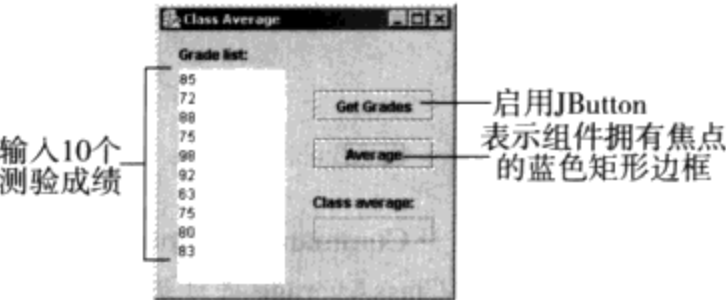


图 9.3 输入 10 个成绩以后的班级平均分应用程序

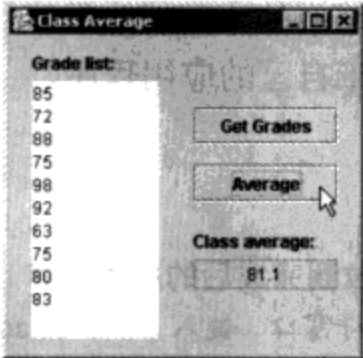


图 9.4 计算出平均值的班级平均分应用程序

6. 输入另一组成绩 在不重新启动应用程序的基础上再计算出另外10个成绩的班级平均分。当点击Get Grades JButton时, 会再次出现一些相应的输入对话框。
7. 关闭正在运行的应用程序 点击关闭按钮关闭正在运行的应用程序。
8. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

9.2 do...while 循环语句

do...while 循环语句类似于 while 语句——当循环-继续条件为 true 时, 相应的每条语句会进行迭代。在 while 语句中, 循环-继续条件是在循环开始时, 即在循环体执行前被测试的。而 do...while 语句则是在循环体执行完成以后才去测试其循环-继续条件的。因此, 在 do...while 语句中, 循环体至少要被执行一次。前面曾经讲过, while 语句是当且仅当其循环-继续条件为 true 时才开始执行的, 因此 while 语句的循环体有可能从不执行。另外, 当 do...while 语句结束时, 将继续执行位于 do...while 语句后的第一条语句。

为了说明 do...while 语句的循环风格, 考虑一个收拾手提箱的例子: 在开始收拾前, 手提箱是空的, 但需要在手提箱中至少放置一件东西。当某个东西被放置在手提箱中以后, 紧接着会去判断手提箱是否已满。只要手提箱不满, 就可继续放置物品(假设本例中需要将尽可能多地物品放置在手提箱中)。下面, 我们再考虑一个 do...while 语句的例子, 请看以下可用于在 JTextArea 中显示出 1~3 的一个应用程序片段:

```
int counter = 1;
do
{
    displayJTextArea.append( counter + "\n" );
    counter++;
}
while ( counter <= 3 );
```



常见编程错误

当 do...while 循环语句的循环-继续条件不会变为 false 时, 会出现一个无限循环。



错误预防提示

确保每一个 do...while 循环体中都包含可使循环-继续条件最终变为 false 的代码。

这段应用程序首先将变量 counter 初始化为 1。其中, do...while 语句中的循环-继续条件为 counter <= 3。在 do...while 语句执行完一次它的循环体后, 会在其循环-继续条件为 true 时进行循环(反复执行)。该 do...while 中包含了一个由两条语句所组成的语句块。第一条语句调用 JTextArea 的 append 方法, 将 counter 的值(还有一个换行符)添加到了 displayJTextArea 中。第二条语句是 counter 值的自增过程。当循环-继续条件变为 false 时(也就是说, 当 counter 大于 3 时), do...while 语句将终止执行而 displayJTextArea 中也将包含从 1 到 3 的数字, 每一个数字会显示在单独的一行上。以下框图将描述该循环语句的每一步执行过程。

执行 do...while 循环语句

1. 应用程序声明变量 counter 并将该值初始化为 1。
2. 应用程序进入 do...while 循环的循环体内。
3. 存储在 counter 中的数(当前值为 1)被添加到了 displayJTextArea 中(包括一个换行符)。
4. 将 counter 的值加 1; 现在 counter 中所存储的值为 2。

5. 检查循环-继续条件。循环-继续条件的计算结果为 true (counter 小于等于 3), 应用程序执行包含在 do...while 循环体内的语句。
6. 存储在 counter 中的数 (当前值为 2) 被添加到了 displayJTextArea 中 (包括一个换行符)。
7. 将 counter 的值加 1; 现在 counter 中所存储的值为 3。
8. 检查循环-继续条件。循环-继续条件的计算结果为 true (counter 小于等于 3), 应用程序执行包含在 do...while 循环体内的语句。存储在 counter 中的数 (当前值为 3) 被添加到了 displayJTextArea 中 (包括一个换行符)。
9. 将 counter 的值加 1; 现在 counter 中所存储的值为 4。
10. 检查循环-继续条件。循环-继续条件的计算结果为 false (counter 不小于等于 3), 应用程序退出 do...while 语句。
11. 继续执行紧接 do...while 语句的下一条语句。

当某循环执行迭代的次数比其应执行的次数多一次或少一次的现象被称为偏一错误 (属于一种逻辑错误)。这种逻辑错误往往是由于提供了不正确的循环-继续条件而引入到应用程序中的。比如, 本节中所讨论的 do...while 语句应该执行三次循环。如果条件被错误地写成了 `counter < 3` 或 `counter <= 2`, 则 JTextArea 中只显示 1 和 2。引起偏一错误最常见的原因是, 在任何循环语句的条件中包含了一个错误的关系运算符 (如 `counter < 3` 中的小于号) 或者是在循环计数器中出现了一个不正确的边界值 (如 `counter <= 2` 中的 2)。

图 9.5 中利用一个 UML 活动图对前面的那个 do...while 语句进行了解释说明。图中清晰地表明, 其循环-继续警戒条件 (`[counter <= 3]`) 至少需要让循环进入到操作状态一次时才会得到检查。前面曾经讲到过, 操作状态可包含一条 Java 语句, 或者更多条相继执行的 (顺序执行的) Java 语句, 如上例。当使用 do...while 语句创建自己的应用程序时, 需要为应用程序提供适当的操作状态与警戒条件。

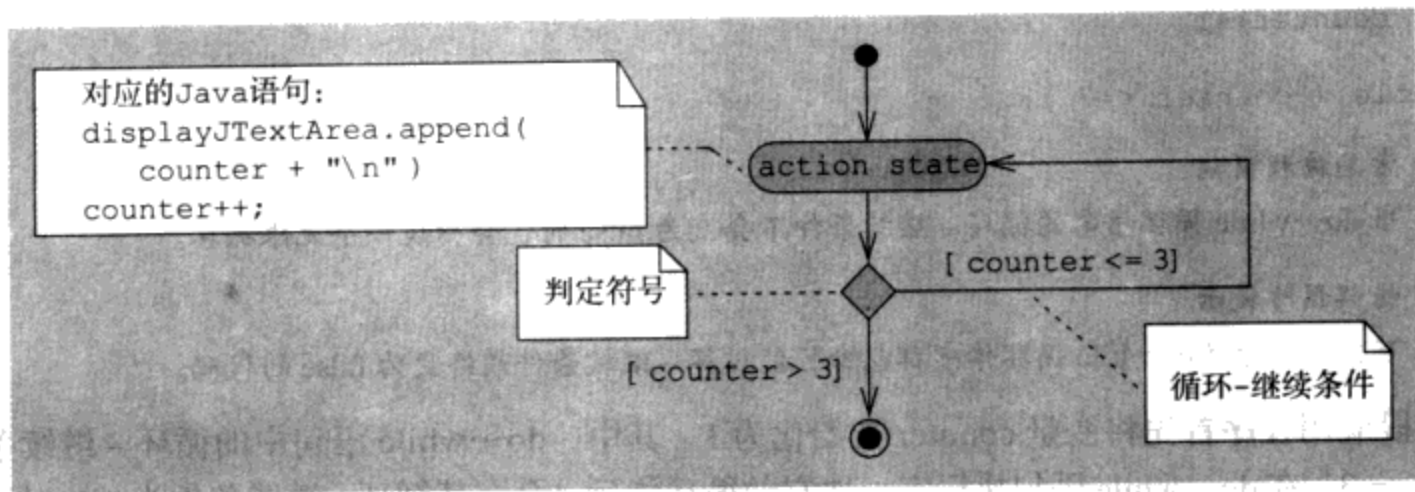


图 9.5 do...while 循环语句的 UML 活动图



错误预防提示

在循环语句的条件中包含一个边界值 (并选择适当的关系运算符) 会减少偏一错误的发生。例如, 在一个打印 1~10 之间值的 do...while 语句中, 循环继续条件应该为 `counter <= 10`, 而不应该是 `counter < 10` (属于一个偏一错误) 或是 `counter < 11` (正确, 但缺乏清晰)。

自测题

1. do...while 语句对循环-继续条件的测试是发生在 ____。
 - a) 循环体执行完毕以后
 - b) 循环体执行完毕以前
 - c) (a)和(b)
 - d) 以上答案均不对
2. 当 do...while 语句中的循环-继续条件 ____ 时, 会出现一个无限循环。
 - a) 不会变为 true
 - b) 不会变为 false
 - c) 为 false
 - d) 被反复测试

答案: 1) a 2) b

9.3 创建班级平均分应用程序

既然我们已经学习了 do...while 语句，下面就准备开发这个属于自己的班级平均分应用程序。首先，将学习相关的列有所执行的操作及表明这些操作执行顺序的伪代码。然后，将采用计数器控制的循环依次地输入成绩。回顾前面曾经提到过，这项技术采用的是一个被称之为计数器的变量来确定语句集合所要执行的次数。在本例中，当 counter 超过 10 时，循环将终止执行。以下便是班级平均分应用程序的伪代码算法：

```
当用户点击 Get Grades JButton 时
    将总成绩设置为 0
    将成绩计数器设置为 1
    清除 JTextArea 中的内容
    清除输出文本框 JTextField 中的内容
Do
    从输入对话框中取得下一个成绩
    将成绩追加至 JTextArea 中
    将成绩累加到总成绩当中
    计数器加 1
    当成绩计数器小于或等于 10
        启用 Average JButton
        将焦点赋予 Average JButton
当用户点击 Average JButton 时
    将总成绩除以 10 计算出班级平均分
    将班级平均分显示在输出 JTextField 中
    禁用 Average JButton
    将焦点赋予 Get Grades JButton
```

我们已对班级平均分应用程序进行了探试并研究了它的伪代码表示，下面，将使用一张 ACE 表，帮助读者最终把这个伪代码转换成 Java 的实现。图 9.6 中列出了该应用程序中相应的操作、组件以及事件，以帮助读者完成属于自己版本的这一应用程序。



	操作	组件	事件
	标记应用程序中的组件	gradeListJLabel, ClassAverageJLabel getGradesJButton	当应用程序运行时
	将总成绩设置为 0		当用户点击 Get Grades JButton 时
	将成绩计数器设置为 1		
	清除 JTextArea 中的内容		
	清除输出文本框 JTextField 中的内容	gradeListJTextArea classAverageJTextField	
	Do		
	从输入对话框中取得下一个成绩	JOptionPane	
	将成绩追加至 JTextArea 中	gradeListJTextArea	
	将成绩累加到总成绩当中		
	计数器加 1		
	当成绩计数器小于或等于 10		当用户点击 Average JButton 时
	启用 Average JButton	averageJButton	
	将焦点赋予 Average JButton	averageJButton	
		averageJButton	
	将总成绩除以 10 计算出班级平均分		
	将班级平均分显示在输出 JTextField 中	ClassAverageJTextField	
	禁用 Average JButton	averageJButton	
	将焦点赋予 Get Grades JButton	getGradesJButton	

图 9.6 班级平均分应用程序的 ACE 表

该应用程序中的 GUI 采用了两个 JLabel (classAverageJLabel 和 gradeListJLabel) 进行标记。用户通过点击 getGradesJButton 输入成绩。该 JButton 的 actionPerformed 事件处理程序将显示出可允许用户输入成绩的输入对话框。所输入的成绩将被加入到总成绩中并显示在 gradeListJTextArea 中。当用户输入完 10 个成绩以后, averageJButton 会被启用。通过点击这一 JButton, 便可使总成绩除以 10, 从而计算出班级的平均分。然后, 班级平均分会在 classAverageJTextField 中得到显示。

既然我们已经阐明了这个解决班级平均分问题的算法, 下面就准备为该模板应用程序添加相应的功能。以下示例将带领读者向 Get Grades JButton 的事件处理程序添加功能, 以便显示出一个输入对话框。

在班级平均分应用程序中输入成绩

1. 将模板复制到工作目录中 将 C:\Examples\Tutorial09\TemplateApplication\ClassAverage 目录复制到 C:\SimplyJava 目录中。
2. 打开班级平均分应用程序的模板文件 在自己的文本编辑器中打开模板文件 ClassAverage.java。
3. 查看总成绩变量 观察模板代码中第 23 行上的声明 (参见图 9.7)。此行声明了一个变量 total 并将其初始化为 0。该变量用于存储所有成绩的总和, 并用于班级平均值的计算之中。读者已经学习了有关事件处理程序中变量及其变量声明的相关知识。在方法中声明的变量被认为是局部变量, 它只能在所声明方法的内部使用。一旦方法执行结束, 其局部变量将不可再被使用。在以前的应用程序中, 局部变量当其所声明的方法执行结束以后, 都不再需要。然而, 变量 total 需在 Get Grades JButton 和 Average JButton 的两个相应事件处理程序的方法中使用。为了能在这两个 JButton 之间点击时, 保留住 total 的值, 此变量需被声明为一个实例变量。实例变量是在一个类的内部定义但又脱离于任何方法。通过在事件处理程序方法的外部声明变量 total, 可以使 total 成为一个可由任何该类方法所访问的实例变量, 包括在 Get Grades 和 Average JButton 的事件处理程序中进行访问。关键字 private 是一个“存取说明符”, 表明变量 total 只能在该类中的方法内进行使用。局部变量与实例变量将在教程 14 中得到进一步的讨论。我们还将再在教程 18 中学习 private 关键字的有关知识。

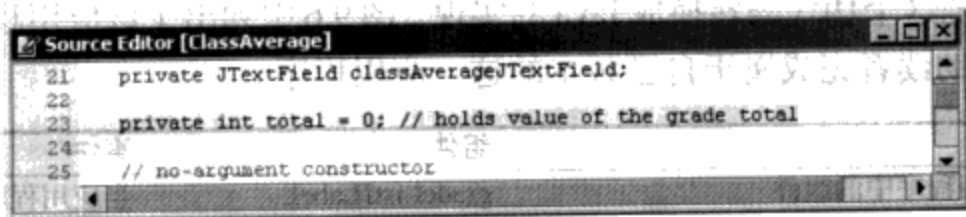


图 9.7 在方法外侧声明的变量 total

4. 声明班级平均分应用程序所使用的变量 将图 9.8 中第 114 行至第 117 行插入到起始于第 112 行的 getGradesJButtonActionPerformed 事件处理程序中。第 114 行是将 total 实例变量的值设置为 0, 以清除任何以前所遗留的总成绩。第 115 行声明并初始化了整数型的局部变量 counter, 它将作为循环语句的计数器。像总成绩和计数器这样的变量, 在使用之前具备适当的初始值是很重要的。否则, 就有可能出现错误。变量 input (在第 116 行声明) 用于存储经输入对话框而输入的用户数据。需将这一输入转换成一个整数, 并存储在局部变量 grade 中 (参见第 117 行)。



错误预防提示

在使用计数器和总成绩之前对其进行初始化, 将有助于防止编译错误和逻辑错误。

5. 将以前任何的计算结果从 Grade list:JTextArea 和 Class Average:JTextField 中清除掉 将图 9.9 中第 119 行至第 121 行插入到 getGradesJButtonActionPerformed 事件处理程序中, 它是在点击 Get Grades JButton 时开始执行的。任何在 JTextArea 和 JTextField 中的文本都应该在下一次计算来临之前得到清除。第 120 行至第 121 行使用 setText 方法清除了这两个组件中的 text 属性。

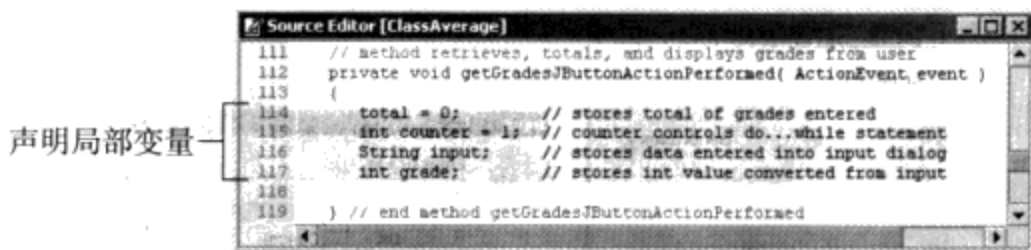


图 9.8 初始化应用程序中的变量

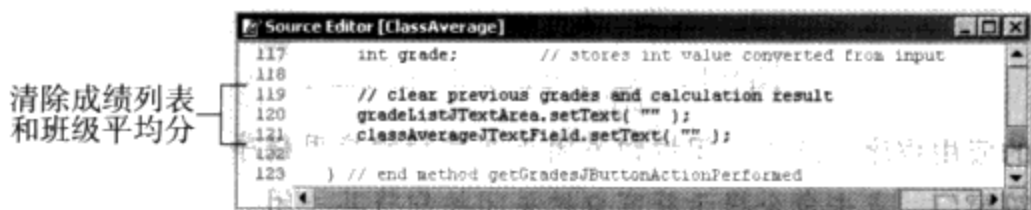


图 9.9 清除输出组件里的内容

6. 显示并取得来自输入对话框的数据 将图 9.10 中第 123 行至第 125 行插入到 `getGradesJButtonActionPerformed` 事件处理程序中。第 124 行通过调用 `JOptionPane.showInputDialog` 方法显示出了一个输入对话框，该方法不但能够显示出一个输入对话框（正如在探试中所看到的）而且还将返回由用户向该对话框的 `JTextField` 中输入的数据。`showInputDialog` 中的第一个参数 `null`，表明该对话框应出现在屏幕的中央位置。第二个参数则指定了位于输入对话框 `JTextField` 之上的文本（有时也称为提示信息）。该对话框会在点击 OK `JButton` 后消失，而输入进对话框 `JTextField` 中的数据将被作为一个 `String` 返回。第 125 行通过调用 `Integer.parseInt` 方法将输入进对话框中的数据转换成了一个整数。最终结果将被存储在 `int` 型的变量 `grade` 中。

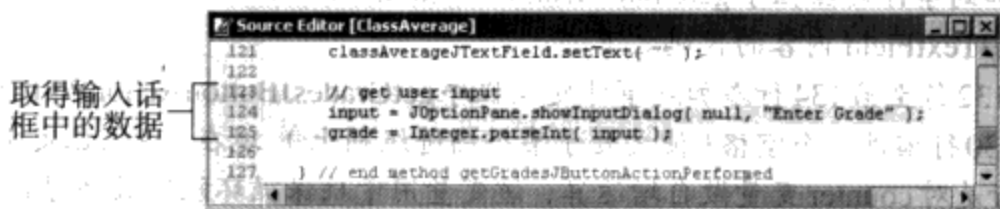
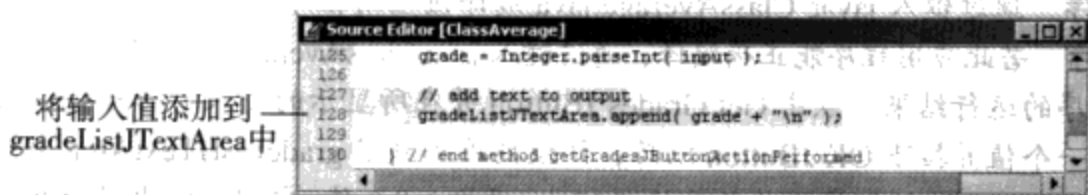


图 9.10 使用一个输入对话框来取得成绩

7. 将数据添加到 `JTextArea` 组件中 将图 9.11 中第 127 行至第 128 行添加到 `getGradesJButtonActionPerformed` 事件处理程序中。第 128 行采用 `JTextArea` 的 `append` 方法，将所输入的成绩和一个换行字符分别添加到 `gradeListJTextArea` 中。

图 9.11 将输入成绩添加到 `gradeListJTextArea` 中

8. 保存应用程序 保存修改后的源代码文件。

9. 打开命令提示符窗口改变目录 选择 `Start → Programs → Accessories → Command Prompt` 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\ClassAverage` 进入到当前的工作目录中。

10. 编译应用程序 通过键入 `javac ClassAverage.java` 编译该应用程序。

11. 运行应用程序 若此应用程序能正确编译，通过键入 `java ClassAverage` 来运行它。图 9.12 中显示了更新后的应用程序的运行结果。应用程序现在可通过每次点击 `Get Grades JButton` 来输入一个成绩（使用一个输入对话框）。在对话框中，输入值 78，然后点击 OK `JButton`。这时，又将返回到应用程序的 `JFrame` 中，而值 78 则会被相应地显示在 `JTextArea` 中。注意，如果再次点击 `Get Grades JButton` 并输入一个新值，那么该值将替换掉（改写）原来的 78。切记，如果 `JTextArea` 中已存有数据，则在事件处理程序开始执行时会清除该项文本。注意观察 `Average JButton` 现在并不可用。

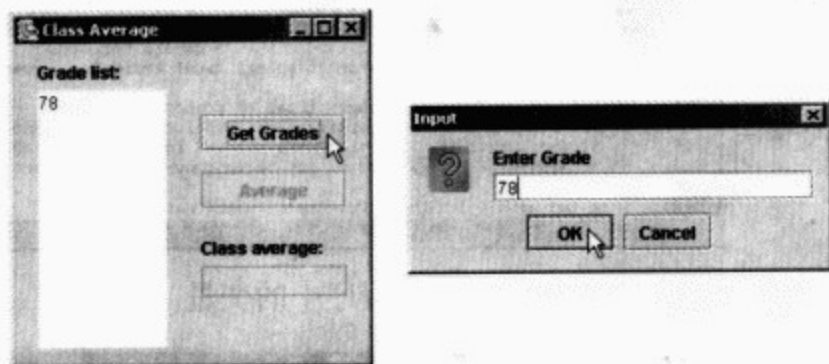


图 9.12 运行更新后的应用程序

12. 关闭正在运行的应用程序 点击关闭按钮关闭正在运行的应用程序。

13. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

到目前为止，我们已添加了当用户点击 Get Grades JButton 时，用于在 JTextArea 中显示经输入对话框而输入的一个成绩的相关代码。下一步，将利用一个 do...while 循环来输入多个成绩。

输入多个成绩

1. 创建一个可输入 10 个成绩的 do...while 循环 每当用户点击 Get Grades JButton 时，该应用程序能够确切地接受 10 个成绩，然后启用 Average JButton，以使用户利用点击的方式计算出班级的平均分。若用户已输入 10 个成绩并再次按下 Get Grades JButton，则现有的 10 个成绩将被清除，并重新要求接受 10 个新成绩。输入对话框会在 Get Grades JButton 点击时，确切地显示出 10 次。将图 9.13 中第 123 行至第 124 行添加到事件处理程序 getGradesJButtonActionPerformed 中，即位于清除 gradeListJTextArea 与 classAverageJTextField 内容的代码之下。

将图 9.13 中第 132 行至第 134 行添加到事件处理程序 getGradesJButtonActionPerformed 的末尾处。应使第 125 行至第 130 行缩进三个空格（使之处于某个缩进级别上）。现在，我们已定义好了一个 do...while 循环。第 132 行是对 counter 变量做自增运算，该变量用于确保循环能够最终停止执行。第 134 行用于判定 counter 是否小于或等于 10。在第 10 个成绩输入以后，第 132 行将把 counter 自增为 11，因而 do...while 语句会终止执行。

2. 保存应用程序 保存修改后的源代码文件。

3. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\ClassAverage 进入到当前的工作目录中。

4. 编译应用程序 通过键入 javac ClassAverage.java 编译该应用程序。

5. 运行应用程序 若此应用程序能正确编译，通过键入 java ClassAverage 来运行它。图 9.14 中显示了更新后的应用程序的运行结果。点击 Get Grades JButton 并在所显示的输入对话框中输入一个值。注意观察在输入完一个值并按下 OK JButton 以后，该值被添加到了 gradeListJTextArea 中，且另一个对话框会立即出现。do...while 循环使操作总共执行 10 次。再输入 9 个值，它们将依次显示在 gradeListJTextArea 中。然而，注意到当 Average JButton 被点击时，班级平均分仍未得到显示。我们将很快对这部分应用程序的功能进行添加。

6. 关闭正在运行的应用程序 点击关闭按钮关闭正在运行的应用程序。

7. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

当用户输入成绩时，应用程序应对这些即将在平均值计算中所使用的成绩，执行求和的运算。当输入完 10 个成绩并将它们显示在 gradeListJTextArea 中以后，应用程序应启用 Average JButton，以使用户能够通过点击它的方式计算并显示出 10 个成绩的平均值。接下来，需要修改循环语句，使得这些成绩在输入时便执行相应的求和运算。然后，还将添加用以启用 Average JButton 的代码，以及在 averageJButtonActionPerformed 事件处理程序中计算平均值并将结果显示在 Class average: JTextField 的代码中。

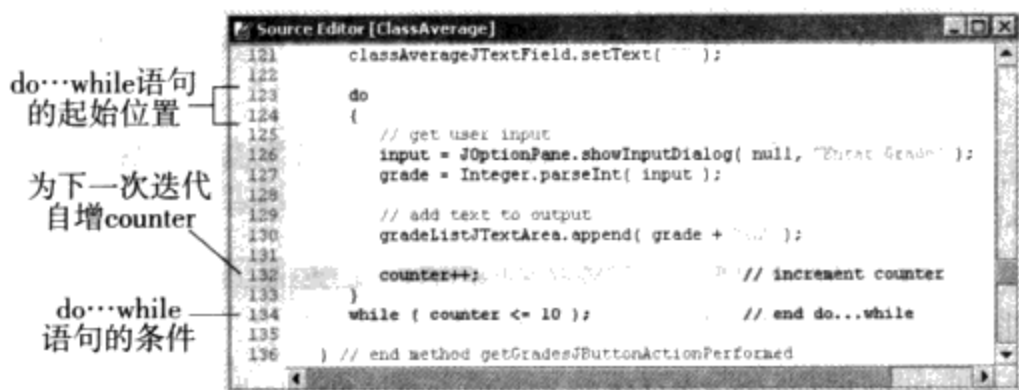


图 9.13 定义 do...while 循环

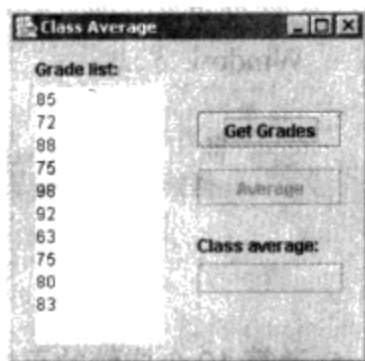


图 9.14 运行更新后的应用程序

计算班级平均分

1. 对在 JTextArea 中显示的成绩进行求和运算 将图 9.15 中第 131 行添加到 getGradesJButtonActionPerformed 事件处理程序中（位于 counter 自增运算的语句之前）。第 131 行是将用户的上一次输入加到变量 total 中。该语句出现在 do...while 循环中，因此它将被执行 10 次。当 do...while 循环终止执行时，total 中将包含所输入的 10 个成绩的总和。

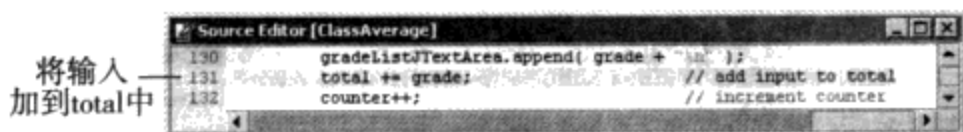


图 9.15 对成绩求和

2. 查看 Average JButton 的初始状态 将屏幕滚动至源代码文件中第 76 行所在的位置处（参见图 9.16）。应用程序不应该允许用户在输入 10 个成绩之前就去计算平均分。为阻止 Average JButton 的事件处理程序中的方法在成绩输入完之前就去执行，第 74 行将利用 setEnabled 方法把 JButton 的 enabled 属性设置为 false，这样便可对这一 JButton 实现禁用。由于 Average JButton 是在应用程序的开始位置处被禁用的，因此当这个 JButton 被点击时，不会执行任何的代码。接着下一步，将把这个 Average JButton 的 enabled 属性设置为 true。

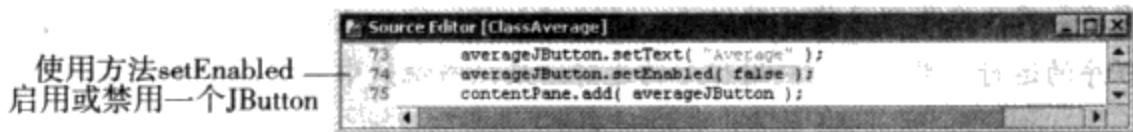


图 9.16 禁用一个 JButton



GUI 设计提示

若 JButton 的功能对用户来说是不可用的，应禁用该 JButton。

3. 启用 Average JButton 将图 9.17 中第 136 行添加到 getGradesJButtonActionPerformed 中。一旦输入了 10 个成绩，用户便可以点击这个 Average JButton 了。第 136 行通过调用 setEnabled 方法并为其传递参数 true 来启用 averageJButton。

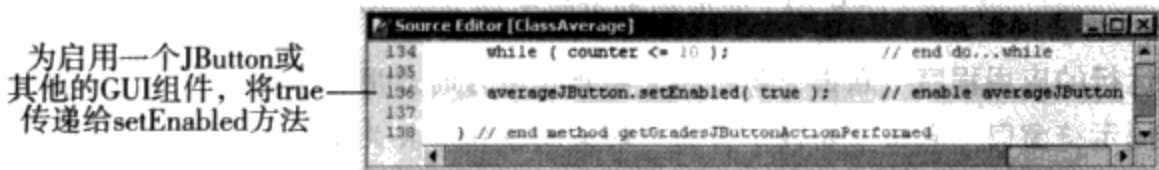


图 9.17 启用一个 JButton



GUI 设计提示

若 JButton 的功能对用户来说是可用的，应启用该 JButton。

4. 将焦点转移至 Average JButton 将图 9.18 中第 137 行添加到 getGradesJButtonActionPerformed 中。用户最有可能是在 Average JButton 被启用以后紧接着就去点击它，因此，可在此时将应用程序的焦点设置

(换句话说,就是转移焦点)到 `averageJButton` 上。为完成这项工作,需调用 `JButton` 的 `requestFocusInWindow` 方法。

使用方法 `requestFocusInWindow`
将焦点转移到某个GUI组件上

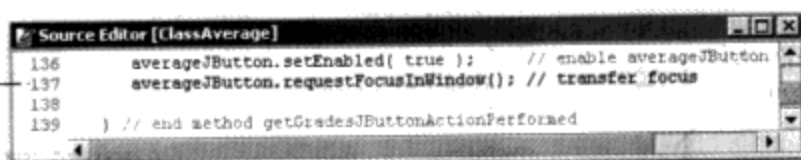


图 9.18 将焦点转移到一个 JButton 上

5. 计算 10 个成绩的平均值 将图 9.19 中第 144 行至第 145 行添加到 `averageJButtonActionPerformed` 中,它会在点击 `Average JButton` 时执行。第 144 行通过将输入的成绩总和除以 10,计算出所需的平均值。回顾 5.5 节曾经提到过,整数间的除法运算其结果仍是一个整数——任何小数部分都将被截去。在这个例子中,我们可能希望有一个更为精确的浮点结果(正如图 9.4 中的结果 81.1)。为了得到这个结果,在除法操作的前面添加一个 `(double)`。在 Java 程序设计中,经常需要将某些类型的名称放置在一个圆括号中(如这里所使用的)。它被认为是一种造型运算符,利用它可将操作数(本例中的 `total`)转换成括号内所需造型的某种类型。第 144 行的除法运算,将所得到的浮点数结果作为一个 `double` 型数,并存储到局部变量 `average` 中。我们将在教程 14 中学到更多的有关造型的知识。第 145 行用于将变量 `average` 的值显示在 `classAverageJTextField` 中。

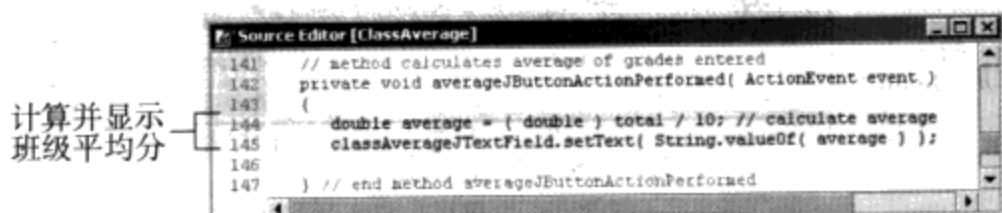


图 9.19 计算并显示班级平均分

6. 保存应用程序 保存修改后的源代码文件。

7. 打开命令提示符窗口改变目录 选择 `Start → Programs → Accessories → Command Prompt` 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\ClassAverage` 进入到当前的工作目录中。

8. 编译应用程序 通过键入 `javac ClassAverage.java` 编译该应用程序。

9. 运行应用程序 若此应用程序能正确编译,通过键入 `java ClassAverage` 来运行它。图 9.20 中显示了完成后的应用程序的运行结果。此时应用程序的运行结果应与探试中的结果完全一样。

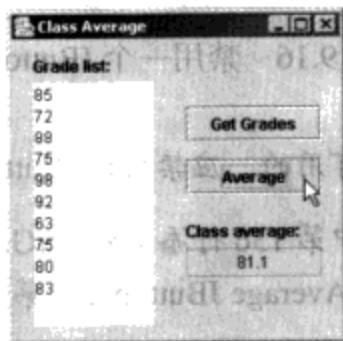


图 9.20 完成后的班级平均分应用程序

10. 关闭正在运行的应用程序 点击关闭按钮关闭正在运行的应用程序。

11. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

图 9.21 中给出了班级平均分应用程序的完整源代码。本教程中,凡需要添加、查看或是修改的代码均在图中相应的代码行中做了突出的显示。

```
1 // Tutorial 9: ClassAverage.java
2 // Application enables user to have the average of grades calculated.
3 import java.awt.*;
```

```

4 import java.awt.event.*;
5 import javax.swing.*;
6
7 public class ClassAverage extends JFrame
8 {
9     // JLabel and JTextArea for list of grades
10    private JLabel gradeListJLabel;
11    private JTextArea gradeListJTextArea;
12
13    // JButton initiates retrieving grades
14    private JButton getGradesJButton;
15
16    // JButton initiates calculating average
17    private JButton averageJButton;
18
19    // JLabel and JTextField used to display average
20    private JLabel classAverageJLabel;
21    private JTextField classAverageJTextField;
22
23    private int total = 0 ; // holds value of the grade total 声明一个实例变量
24
25    // no-argument constructor
26    public ClassAverage()
27    {
28        createUserInterface();
29    }
30
31    // create and position GUI components; register event handlers
32    private void createUserInterface()
33    {
34        // get content pane for attaching GUI components
35        Container contentPane = getContentPane();
36
37        // enable explicit positioning of GUI components
38        contentPane.setLayout( null );
39
40        // set up gradeListJLabel
41        gradeListJLabel = new JLabel();
42        gradeListJLabel.setBounds( 16 , 8 , 70, 23 );
43        gradeListJLabel.setText( "Grade list:" );
44        contentPane.add( gradeListJLabel );
45
46        // set up gradeListJTextArea
47        gradeListJTextArea = new JTextArea();
48        gradeListJTextArea.setBounds( 16 , 32, 88, 180 );
49        contentPane.add( gradeListJTextArea );
50
51        // set up getGradesJButton
52        getGradesJButton = new JButton();
53        getGradesJButton.setBounds( 128 , 50 , 100, 26 );
54        getGradesJButton.setText( "Get Grades" );
55        contentPane.add( getGradesJButton );
56        getGradesJButton.addActionListener(
57
58            new ActionListener() // anonymous inner class
59            {
60                // event handler called when getGradesJButton is clicked
61                public void actionPerformed( ActionEvent event )
62                {
63                    getGradesJButtonActionPerformed( event );
64                }
65            }
66        );

```

```

65
66     } // end anonymous inner class
67
68     ); // end call to addActionListener
69
70     // set up averageJButton
71     averageJButton = new JButton();
72     averageJButton.setBounds( 128 , 90 , 100, 26 );
73     averageJButton.setText( "Average" );
74     averageJButton.setEnabled( false ); // 禁用Average JButton
75     contentPane.add( averageJButton );
76     averageJButton.addActionListener(
77
78         new ActionListener() // anonymous inner class
79         {
80             // event handler called when averageJButton is clicked
81             public void actionPerformed((ActionEvent event) )
82             {
83                 averageJButtonActionPerformed( event );
84             }
85
86         } // end anonymous inner class
87
88     ); // end call to addActionListener
89
90     // set up classAverageJLabel
91     classAverageJLabel = new JLabel();
92     classAverageJLabel.setBounds( 128 , 132, 90 , 23 );
93     classAverageJLabel.setText( "Class average:" );
94     contentPane.add( classAverageJLabel );
95
96     // set up classAverageJTextField
97     classAverageJTextField = new JTextField();
98     classAverageJTextField.setBounds( 128 , 156 , 100, 21 );
99     classAverageJTextField.setEditable( false );
100    classAverageJTextField.setHorizontalAlignment(
101        JTextField.CENTER );
102    contentPane.add( classAverageJTextField );
103
104    // set properties of application's window
105    setTitle( "Class Average" ); // set title bar text
106    setSize( 250 , 250 ); // set window size
107    setVisible( true ); // display window
108
109 } // end method createUserInterface
110
111 // method retrieves, totals and displays grades from user
112 private void getGradesJButtonActionPerformed( ActionEvent event )
113 {
114     total = 0 ; // stores total of grades entered
115     int counter = 1 ; // counter controls do...while statement // 声明局部变量
116     String input; // stores data entered into input dialog
117     int grade; // stores int value converted from input
118
119     // clear previous grades and calculation result // 清除 JTextArea 和
120     gradeListJTextArea.setText( "" ); // JTextField 中的内容
121     classAverageJTextField.setText( "" );
122
123     do
124     {
125         // get user input

```

```

126     input = JOptionPane.showInputDialog( null, "Enter Grade" );
127     grade = Integer.parseInt( input );
128                                     利用 do...while 语句计算班级平均分
129     // add text to output
130     gradeListJTextArea.append( grade + "\n" );
131     total += grade;                // add input to total
132     counter++;                    // increment counter
133 }
134 while ( counter <= 10 );        // end do...while
135
136     averageJButton.setEnabled( true );    // enable averageJButton
137     averageJButton.requestFocusInWindow(); // transfer focus
138                                     启用 Average JButton
139 } // end method getGradesJButtonActionPerformed    并赋予焦点
140
141 // method calculates average of grades entered
142 private void averageJButtonActionPerformed((ActionEvent event) )
143 {
144     double average = ( double ) total / 10 ; // calculate average 计算并显示
145     classAverageJTextField.setText( String.valueOf( average ) ); 班级平均分
146
147 } // end method averageJButtonActionPerformed
148
149 // main method
150 public static void main( String[] args )
151 {
152     ClassAverage application = new ClassAverage();
153     application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
154
155 } // end method main
156
157 } // end class ClassAverage

```

图 9.21 班级平均分应用程序代码

自测题

1. _____ 运算符可将其操作数转化为由圆括号中所指定的类型。

- a) 类型 b) 转换器 c) 转换 d) 造型

2. 调用 `JOptionPane` 的 _____ 方法可显示一个输入对话框。

- a) `showInput` b) `showMessageDialog` c) `showInputDialog` d) 以上答案均不对

答案: 1) d 2) c

9.4 小结

本教程中, 读者学习了如何使用 `do...while` 循环语句。并对一个可作为解释这种语句执行的 UML 活动图进行了研究。我们已在自己的班级平均分应用程序中使用到了这个 `do...while` 语句。同时, 还学习了在类中多个方法之间如何存储所用信息的一种变量——实例变量。

`do...while` 循环语句, 只要其循环-继续条件为 `true`, 就将继续执行。这种循环语句总是至少执行一次它的循环体, 只有当循环-继续条件变为 `false` 时, 才会终止其循环的执行。

另外, 我们还学习了如何使用输入对话框从用户那里获取输入。了解了如何启用和禁用一个 `JButton`, 以及如何将应用程序的焦点转移到一个 `JButton` 上。除此之外, 还学习了如何使用一个造型运算符在不同的数值类型间进行转换。在下一个教程中, 将继续研究循环语句, 学习如何使用 `for` 循环语句, 它在计数器控制的循环中同样是特别有用的。

技术小结

使用 do...while 循环语句

- 在代码中插入 do...while 循环语句并把希望至少执行一次的一些语句放置在它的循环体内。该循环在其循环 - 继续条件保持为 true 时, 会一直迭代下去。
- 将循环 - 继续条件放置在位于 while 循环体末尾之后的圆括号内。
- 将一个分号 (;) 放置在位于封闭其条件的右圆括号的后面。

显示一个输入对话框

- 调用 JOptionPane.showInputDialog 方法。
- 使用 null 作为第一个参数, 用于将对话框居中到屏幕上。
- 将对话框中所要显示的文本 (提示信息) 指定为第二个参数。

从输入对话框中获取输入

- 输入对话框中的输入会被 JOptionPane.showInputDialog 方法返回为一个 String。

禁用一个 JButton

- 调用 setEnabled 方法, 并为其传递参数 false。

启用一个 JButton

- 调用 setEnabled 方法, 并为其传递参数 true。

将焦点转移到某个组件上

- 调用该组件的 requestFocusInWindow 方法。

声明一个实例变量

- 在某个应用程序类中声明的但又处于任何方法之外的一种变量。

将一个 int 型变量的值转化为一个 double

- 将造型运算符 (double) 置于整型变量名之前。

关键技术语

造型运算符 可将其操作数转换为造型圆括号中所放置的类型。

被禁用 某个组件, 其 enabled 属性被设置成为 false。这种组件不响应用户的交互。

do...while 循环语句 一种在循环 - 继续条件为 true 时, 将维持执行其循环体的控制语句。其中, 对条件的测试发生在循环体执行完之后进行的, 因此, 循环体语句总是至少执行一次。

enabled 属性 指定某个组件, 如一个 JButton, 其形态是否是启用的 (true) 或是禁用的 (false)。

焦点 当某个组件被选取时, 它被认为拥有了应用程序的焦点。焦点有助于将用户的注意力转移到下一个即将使用的组件上。

输入对话框 一种可导致应用程序等待用户输入数据的对话框。该对话框中包含了一个用于取得用户输入的 JTextField。

实例变量 一种在类中定义但又位于该类任何方法之外的变量。实例变量的值可在方法之间的调用中得到保留, 因而能够在类中多个方法的内部之间进行使用。

JOptionPane.showInputDialog 方法 用于显示出一个输入对话框。

局部变量 一种只用于所声明方法内部的变量。

偏一错误 对于某个循环, 其执行的迭代次数比所希望的次数多一次或少一次的一种逻辑错误。

requestFocusInWindow 方法 一个可用于设置组件的焦点以吸引用户注意力的方法。

setEnabled 方法 可将某个组件的 enabled 属性设置为 true 或 false 的方法。若需要组件响应用户的交互, 则将 enabled 属性设置为 true。若不需要组件响应用户的交互, 则将 enabled 属性设置为 false。

转移焦点 设置组件使之拥有应用程序的焦点。这是通过调用组件的 requestFocusInWindow 方法来实现的。

GUI 设计导航

JButton

- 若 JButton 的功能对用户来说是不可用的，应禁用该 JButton。
- 若 JButton 的功能对用户来说是可用的，应启用该 JButton。

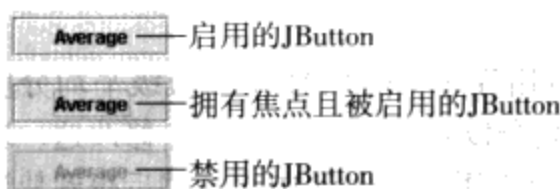
JTextArea

- 大多数的 JTextArea 都应配有一个描述其性质的 JLabel，用于指明所要显示的输出信息。为这样的 JLabel 使用语句大写形式。

Java 类库索引

JButton 该组件可允许用户命令应用程序完成一项操作。

● 运行



● 方法

requestFocusInWindow 设置 JButton 使之拥有焦点。

setBounds 设置 JButton 的边界属性，确定其位置与大小。

setEnabled 将 JButton 的 enabled 属性设置为 true 或 false。

setText 设置 JButton 的 text 属性。

JOptionPane 该类可显示出信息对话框和输入对话框。

● 方法

showInputDialog 显示一个输入对话框。

showMessageDialog 显示一个消息对话框。

习题

选择题

- 9.1 当一个 do...while 中的循环-继续条件不会变为 false 时，会出现_____。
a) 无限循环 b) 计数器控制的循环 c) 控制语句 d) 嵌套的控制语句
- 9.2 _____语句将至少执行一次它的语句体，且能够继续执行下去直至其循环-继续条件变为 false 时为止。
a) while b) if c) do...while d) if...else
- 9.3 JOptionPane.showInputDialog 方法中的第_____个参数用于指定位于输入对话框 JTextField 之上的提示信息。
a) 1 b) 2 c) 3 d) 4
- 9.4 某循环其执行的迭代次数比所希望的次数多一次或少一次的现象被称为_____。
a) 无限循环 b) 计数器控制的循环 c) 偏一错误 d) 嵌套的控制语句
- 9.5 如果循环-继续条件在首次计算中就为 false，则接下来的 do...while 循环语句_____。
a) 执行其循环体直到条件变为 true b) 执行其循环体直到条件变为 false
c) 不会执行 d) 对循环体只执行了一次
- 9.6 由方法 showInputDialog 所显示的输入对话框，含有一个向用户提供输入的_____组件。
a) JTextArea b) JTextField c) JLabel d) 标题栏
- 9.7 可使用方法_____启用（或禁用）一个 JButton 组件。

- a) enable b) requestEnabled c) setEnabled d) requestFocusInWindow
- 9.8 当某个 JButton 拥有了应用程序的焦点时, 可通过点击这个 JButton 或按下 _____ 来执行该 JButton 的操作。
- a) 空格键 b) Tab 键 c) Enter 键 d) 以上答案均不对
- 9.9 方法 showInputDialog 将返回 _____。
- a) 对话框内 JTextField 里的值 b) true —— 当输入值大于 0 时, 否则为 false
c) 什么也没有 d) 0 —— 若用户点击 OK JButton, 否则为 1
- 9.10 在一个 do...while 语句中, 关键字 while 出现在 _____。
- a) do 关键字之后, do...while 循环体之前
b) do 关键字之后, 并作为循环体中的最后一条语句
c) do 关键字之前
d) do 关键字之后, 且 do...while 循环体之后

练习题

- 9.11 (修改后的班级平均分应用程序) 图 9.21 中的班级平均分应用程序有一个不足之处: Average JButton 在首次计算出平均分以后仍处于启用状态。修改这一班级平均分应用程序, 如图 9.22 所示, 使 Get Grades JButton 能够在用户输入完 10 个成绩后得到禁用。这时, 用户的惟一选项便是点击 Average JButton, 以显示出平均分。一旦平均分被显示出来以后, 让应用程序再去禁用 Average JButton, 而去启用 Get Grades JButton, 并将应用程序的焦点赋予 Get Grades JButton, 这样, 用户便能重新输入 10 个新成绩了。
- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial09\Exercises\ModifiedClassAverage 目录复制到 C:\SimplyJava 目录中。
- b) 打开模板文件 在自己的文本编辑器中打开 ClassAverage.java 文件。
- c) 修改 getGradesJButtonActionPerformed 事件处理程序 在第 139 行至第 140 行添加用以禁用 Get Grades JButton 的代码。第 139 行是一个注释行, 第 140 行则用以完成禁用的操作。
- d) 修改 averageJButtonActionPerformed 事件处理程序 从第 150 行开始, 添加一些代码, 用以禁用 Average JButton 及启用 Get Grades JButton, 并将应用程序的焦点赋予 Get Grades JButton。
- e) 保存应用程序 保存修改后的源代码文件。
- f) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\ModifiedClassAverage 进入到当前的工作目录中。
- g) 编译应用程序 通过键入 javac ClassAverage.java, 编译该应用程序。
- h) 运行完成后的应用程序 若应用程序能够正确编译, 键入 java ClassAverage 来运行它。通过点击 Get Grades JButton 及输入 10 个成绩, 测试该应用程序。确信一旦输入 10 个成绩, Get Grades JButton 便被禁用, 而 Average JButton 则被得到启用。另外, 还应为 Average JButton 赋予焦点。然后, 通过按下空格键计算出相应的平均分。确保平均分是正确的。Average JButton 现在应变为禁用的状态, 而 Get Grades JButton 则被启用并拥有焦点。
- i) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- j) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。
- 9.12 (可处理任意数目成绩的班级平均分应用程序) 修改最初的班级平均分应用程序, 使之能处理任意数目的成绩, 如图 9.23 所示。当用户点击 Get Grades JButton 时, 出现一个输入对话框并要求用户输入所要键入的成绩数目。该应用程序的执行过程与探试中的班级平均分应用程序相类似 (参见 9.1 节), 除了每次点击 Get Grades JButton 时, 所出现输入对话框的次数是用户所希望键入的成绩数目。
- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial09\Exercises\ClassAverageAnyNumberOfGrades 目录复制到 C:\SimplyJava 目录中。
- b) 打开模板文件 在自己的文本编辑器中打开 ClassAverage.java 文件。

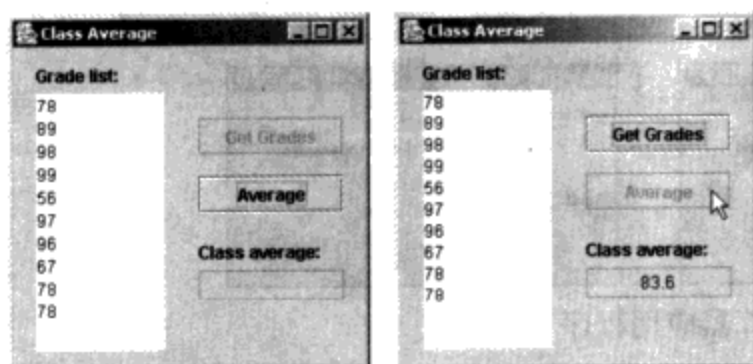


图 9.22 修改后的班级平均分应用程序

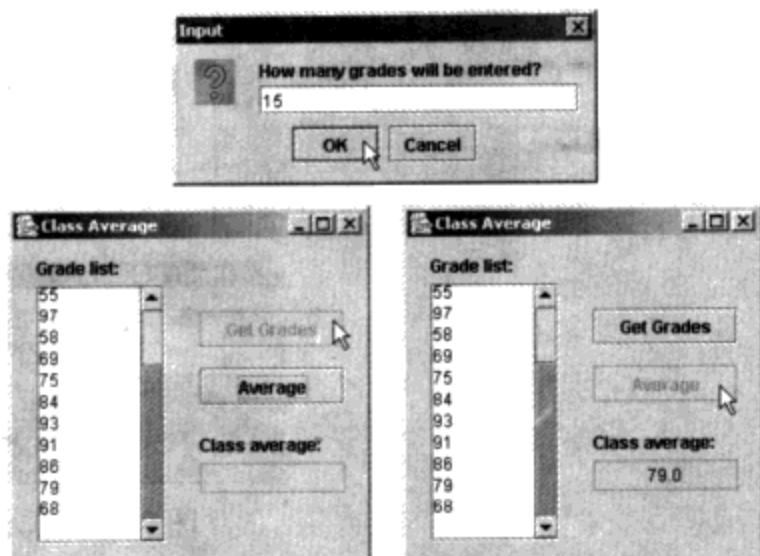


图 9.23 可处理任意数目成绩的班级平均分应用程序

- c) **声明一个实例变量** 在第 26 行添加一个用于声明 `int` 型变量 `limit` 的代码。该变量将存储用户所希望键入的成绩数目。
- d) **修改 `getGradesJButtonActionPerformed` 事件处理程序** 在第 130 行至第 131 行(位于 `do...while` 语句之前)添加作为向用户显示一个输入对话框的代码。使用两行语句进行实现是为了增强代码的可读性。把对话框中所显示的文本设置为 "How many grades will be entered?". 再将这个输入对话框的结果赋值给变量 `input`。在第 132 行添加用于将 `input` 的值转换为一个整数并将结果赋值给实例变量 `limit` 的代码。
- e) **修改 `getGradesJButtonActionPerformed` 中的 `do...while` 语句** 修改 `do...while` 语句的条件(参见 145 行), 使循环语句能够循环由用户所输入的次數。在这个循环条件中, 使用 `limit` 实例变量确定出何时终止循环。
- f) **计算班级平均分** 修改 `averageJButtonActionPerformed` 事件处理程序(参见第 157 行至第 170 行)中的代码, 以使第 158 行能够通过使用实例变量 `limit` 而不是值 10 来计算 `average`。
- g) **保存应用程序** 保存修改后的源代码文件。
- h) **打开命令提示符窗口改变目录** 选择 `Start → Programs → Accessories → Command Prompt` 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\ClassAverageAnyNumberOfGrades` 进入到当前的工作目录中。
- i) **编译应用程序** 通过键入 `javac ClassAverage.java`, 编译该应用程序。
- j) **运行完成后的应用程序** 若应用程序能正确编译, 键入 `java ClassAverage` 来运行它。通过按下 `Get Grades JButton` 测试该应用程序。在首个输入对话框中, 输入所期望的成绩数目。输入对话框出现的次数会与这里所决定输入的成绩数目相同。当所有成绩被输入以后, `Average JButton` 应被启用并拥有焦点。按下空格键, 计算出平均分并确保计算的平均分是正确的。
- k) **关闭应用程序** 点击关闭按钮关闭正在运行的应用程序。

- l) **关闭命令提示符窗口** 点击命令提示符窗口的关闭按钮将其关闭。

9.13 (算术计算器应用程序) 编写一个应用程序, 允许用户输入两个可进行相加或相乘运算的数字(参见图 9.24)。用户应在相应的输入对话框中输入每一个数, 该输入对话框是在 `Enter Operands JButton` 被点击时开始显示的。每个数字都将附加到 `Operands:JTextArea` 中。`Add` 和 `Multiply JButton` 在初始时是被禁用的, 但它们应在输入两个操作数以后得到启用。一旦使用 `Add JButton` 计算出某个结果, 则这一 `JButton` 应被禁用直到重新加入两个数字时为止。

- a) **将模板复制到工作目录中** 将 `C:\Examples\Tutorial09\Exercises\ArithmeticCalculator` 目录复制到 `C:\SimplyJava` 目录中。
- b) **打开模板文件** 在自己的文本编辑器中打开 `ArithmeticCalculator.java` 文件。模板中第 25 行至第 26 行已声明了两个将在应用程序中使用的 `double` 型实例变量——`value1` 和 `value2`。变量 `value1` 用于存储用户输入的第一个值, 而变量 `value2` 则将用于存储用户输入的第二个值。每个变量均被初始化为 0。

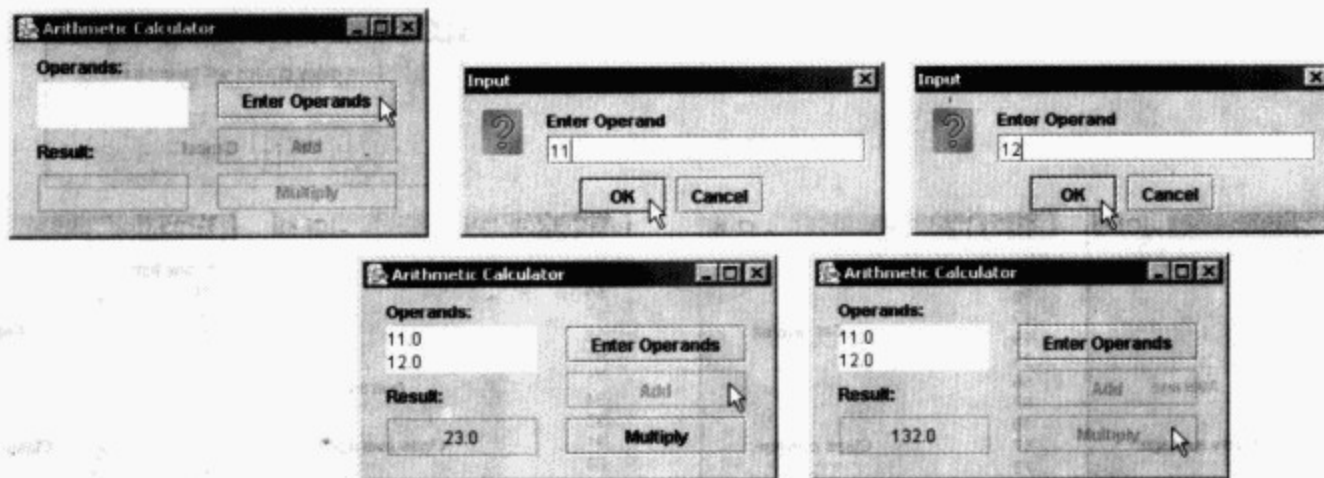


图 9.24 算术计算器应用程序

- c) **显示一个输入对话框** 模板中的 `enterOperandsJButtonActionPerformed` 事件处理程序里已包含了一个起始于第 145 行的 `do...while` 语句。在第 147 行至第 148 行（位于 `do...while` 语句的内部）添加一个要求用户输入一操作数的输入对话框。使用两行语句进行实现是为了增强程序的可读性。将用户在输入对话框中所输入的值存储到变量 `input`（在第 139 行声明）中。
- d) **获取用户的输入** 在所添加的用于显示一个输入对话框代码的后面，会有一个空的 `if...else` 语句。其中，`if` 部分是当 `counter` 中的值为 1 时执行的（也就是说，用户已输入第一个操作数）。在 `if` 语句体中添加一行代码，将 `input` 的值转换为 `double` 类型，并将该值赋予实例变量 `value1`。`else` 部分是当用户已输入了第二个操作数的时候开始执行的。在 `else` 部分中添加一行代码，将 `input` 的值转换为 `double` 类型，并将该值赋予实例变量 `value2`。
- e) **自增循环计数器并修改 `do...while` 语句的条件** 在第 160 行（位于 `do...while` 语句的内部）添加用于自增 `counter` 值的代码。修改该循环语句中位于第 126 行的条件，使 `do...while` 语句能够执行两次。
- f) **将焦点转移至 Add JButton** 在第 171 行至第 172 行，添加可将焦点转移至 Add JButton 的代码。第 171 行应为一个注释。
- g) **修改 `addJButtonActionPerformed` 事件处理程序** 该方法已完成对 `value1` 和 `value2` 执行加法的运算及将结果赋值给 `result` 和在 `resultJTextField` 中显示出这一结果的各项操作。现在，需要禁用这个 Add JButton。将完成此项工作的代码添加到第 182 行至第 183 行。第 182 行作为一个注释来使用，第 183 行则用于禁用该 JButton。
- h) **修改 `multiplyJButtonActionPerformed` 事件处理程序** 该方法已完成对 `value1` 和 `value2` 执行相乘的运算及将结果赋值给 `result` 和在 `resultJTextField` 中显示出这一结果的各项操作。现在需要禁用这个 Multiply JButton。将完成此工作的代码添加到第 193 行至第 194 行。第 193 行作为一个注释来使用，第 194 行则用于禁用该 JButton。
- i) **保存应用程序** 保存修改后的源代码文件。
- j) **打开命令提示符窗口改变目录** 选择 `Start → Programs → Accessories → Command Prompt` 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\ArithmeticCalculator` 进入到当前的工作目录中。
- k) **编译应用程序** 通过键入 `javac ArithmeticCalculator.java`，编译该应用程序。
- l) **运行完成后的应用程序** 若应用程序能正确编译，键入 `java ArithmeticCalculator` 来运行它。通过键入两个输入值并按下 Add 和 Multiply JButton，测试该应用程序的执行情况。确保 Add JButton 在两个数字输入以后将拥有焦点。同时，还应确保任何一个 Add 或 Multiply JButton 在点击时，其加法运算或乘法运算能够得到正确执行且相应的 JButton 也将变为禁用的状态。
- m) **关闭应用程序** 点击关闭按钮关闭正在运行的应用程序。
- n) **关闭命令提示符窗口** 点击命令提示符窗口的关闭按钮将其关闭。

9.14 （说出这段代码的作用）下列代码的运行结果是什么？假设代码在开始执行时，`displayJTextArea` 中并不存在任何的文本。

```

1 int y;
2 int x = 1 ;
3
4 do {
5     y = x * x;
6     displayJTextArea.append( y + "\n" );
7     x += 1 ;
8 } while ( x <= 10 );

```

9.15 (找出代码中的错误) 寻找下列代码中的错误。这段代码应按照递减的顺序将10到1之间的数字附加到 outputJTextArea 中。

```

1 int counter = 10 ;
2
3 do
4 {
5     outputJTextArea.append( counter + "\n" );
6 }
7 while ( counter >= 10 );
8
9 --counter;

```

9.16 (使用调试程序)(阶乘应用程序) 阶乘应用程序用于计算某个由用户输入的整数的阶乘。所谓整数阶乘是指, 将1至某个整数之间的每一个数进行相乘以得的一个乘积的运算。例如, 3的阶乘(按照数学形式可表示为 $3!$)等于6。在这一应用程序的测试过程中, 将注意到它并不能正确执行。利用调试程序寻找并改正应用程序中的逻辑错误。图 9.25 中显示的是该阶乘应用程序的正确结果。

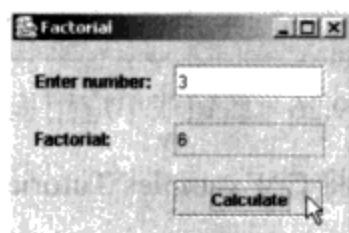


图 9.25 阶乘应用程序的正确结果

- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial09\Exercises\Debugger\Factorial 目录复制到 C:\SimplyJava 目录中。
- b) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\Debugger\Factorial` 进入到当前的工作目录中。
- c) 运行应用程序 通过键入 `java Factorial` 运行阶乘应用程序。在 Enter number:JTextField 中输入 3 并按下 Calculate JButton。注意观察 Factorial:JTextField 中所显示的结果(0)并不是一个正确的值(6)。
- d) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- e) 利用 -g 选项进行编译 为进行调试, 通过键入 `javac -g Factorial.java` 编译应用程序。
- f) 启动调试程序 通过键入 `jdb` 启动调试程序。
- g) 打开模板文件 在自己的文本编辑器中打开 Factorial.java 文件。
- h) 寻找并更正错误 利用前面教程中学到的调试技巧, 定位 `calculateJButtonActionPerformed` 事件处理程序(位于模板代码第 85 行至第 104 行)中的逻辑错误。在 `do...while` 语句中设置一个断点, 以便考察变量 `counter` 的值, 然后, 利用调试程序中的 `print` 命令, 考察每一次循环迭代过程中变量 `counter` 的值。在定位到逻辑错误以后, 修改 `calculateJButtonActionPerformed` 中代码, 使之显示出正确的结果。
- i) 保存应用程序 保存修改后的源代码文件。
- j) 编译应用程序 通过键入 `javac Factorial.java`, 编译该应用程序。

- k) 运行完成后的应用程序 若此应用程序能正确编译, 键入 `java Factorial` 来运行它。测试这一应用程序, 通过使用不同的输入以确保每个输入值都能得到正确的阶乘结果。
- l) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- m) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

挑战题

9.17 (餐馆账单应用程序) 开发一个用于计算某餐馆账单的应用程序 (参见图 9.26)。当用户点击 `Add Items JButton` 时, 用户应输入所选定的项目、所选定项目的数量以及项目的价格。为完成这些操作, 应为每个项目提供三个输入对话框。用户只能输入三项。一旦这三个项目输入完毕, 应用程序应在三个 `JTextArea` 中显示出所有已选定的数量、选定的项目以及每种项目的单价。相应的事件处理程序应计算并显示出总价格。对于每一套输入, 项目信息将被附加到 `JTextArea` 中。另外, 所有价格都应按照美元的格式进行显示 (即数字前应有一个美元符号且小数点的右侧需有两位数字)。注意, 每个输入对话框中的文本会对用户所操作的当前项目显示出一个编号。

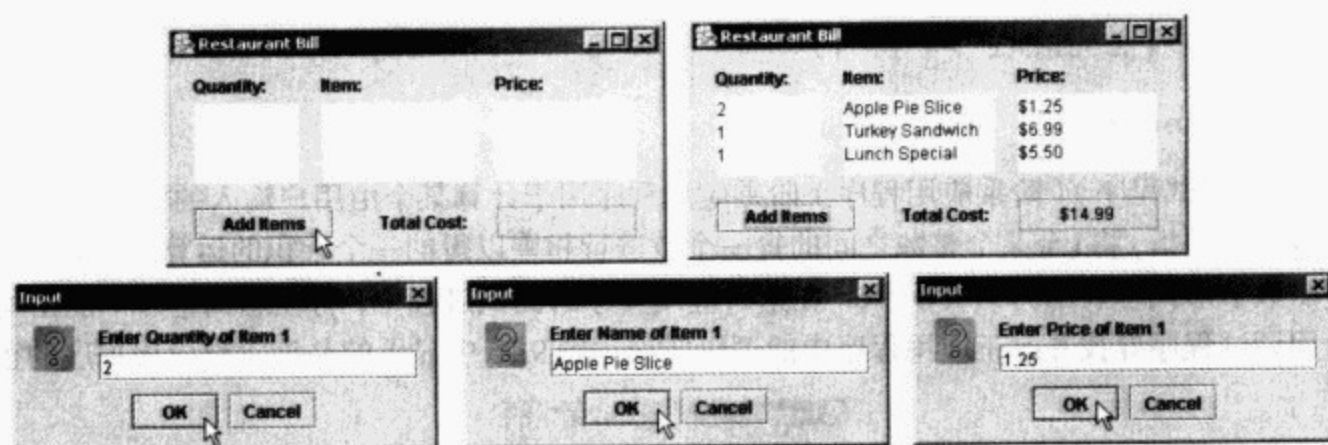


图 9.26 餐馆账单应用程序

- a) 将模板复制到工作目录中 将 `C:\Examples\Tutorial09\Exercises\RestaurantBill` 目录复制到 `C:\SimplyJava` 目录中。
- b) 打开模板文件 在自己的文本编辑器中打开 `RestaurantBill.java` 文件。
- c) 向 `addJButtonActionPerformed` 事件处理程序添加 `do...while` 循环 从第 130 行开始, 添加一条可循环三次的 `do...while` 语句。使用一个被称之为 `counter` 的 `int` 型变量来控制循环执行的次数。
- d) 向 `do...while` 循环添加代码 在 `do...while` 循环的内部, 编写用于显示出三个输入对话框的语句 (使用 `JOptionPane.showInputDialog` 方法), 分别取得某个项目的数量、名称和价格。在输入对话框提示信息中, 利用变量 `counter` 的值, 向用户指示正在为第 1, 第 2 或第 3 套项目输入信息。将用于输入数量的输入对话框中的结果转换成 `int` 类型, 再将用于输入价格的输入对话框中的结果转化成 `double` 类型。使用 `JTextArea` 的方法 `append` 将每个输入对话框所返回的结果附加到相应的 `JTextArea` 中。然后, 通过为当前项目的数量和价格做一个相乘的运算并把结果加入到 `total` 中, 以最终计算出所需的 `total`。最后, 确保 `counter` 的自增运算发生在 `do...while` 循环结束之前。
- e) 保存应用程序 保存修改后的源代码文件。
- f) 打开命令提示符窗口改变目录 选择 `Start → Programs → Accessories → Command Prompt` 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\RestaurantBill` 进入到当前的工作目录中。
- g) 编译应用程序 通过键入 `javac RestaurantBill.java`, 对应用程序进行编译。
- h) 运行完成后的应用程序 若此应用程序能正确编译, 键入 `java RestaurantBill` 来运行它。通过使用一些不同的输入来测试应用程序, 保证每个得到的总价钱都是正确的。同时, 还应确保所有输入都能显示在正确的 `JTextArea` 中。
- i) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- j) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。



教程 10 利息计算器应用程序

介绍 for 循环语句

教学目标

在本教程中，读者将学到以下内容：

- 用来反复执行某些语句的 for 循环
- 在特定范围内获取输入的 JSpinner 组件
- 如何将 JTextArea 放置到 JScrollPane 中以增添滚动条

正如读者在教程8和教程9中学到的，通常情况下，需要应用程序反复执行某些操作。利用while或do...while语句，允许开发人员指定一个循环-继续条件，并在进入该循环之前或者在执行完此循环语句体之后对这一条件进行测试。在汽车贷款计算器应用程序和班级平均分应用程序中，通过使用一个计数器确定出循环应迭代的次数。计数器控制的循环在应用程序中是很常见的，因而Java又提供了另外一种专门用于此目的控制语句——for循环语句。在本教程中，我们将使用for循环语句创建利息计算器应用程序。

10.1 探试利息计算器应用程序

利息计算器应用程序，根据储蓄存款账户下的初始资金、利息率及该资金的存储年限，便可计算出总的资金额。为计算利息，用户需指定本金（该账户下的初始资金）、年利率及年限。应用程序应显示出每年年底该项资金将拥有的总数额（本金+利息）。该应用程序必须满足下面的需求：

应用程序需求分析

某客户考虑投资 \$1000.00 用于储蓄存款（利率为 5%），并希望能预测出整个投资的走势。假设所有利息也将作为存款的一部分进行使用，开发一个应用程序，计算并打印出该账户资金经过 n 年的时间期限，每年年末应得到的收益总数额。为计算相应的数据，可以使用下列公式：

$$a = p(1 + r)^n$$

其中：

p 为初始资金（本金）

r 为年利率

n 为存储年限

a 为第 n 年年末的存款总额。

（注意：为了在计算中正确使用，代表百分比的数需除以 100。比如，在利息计算中，5% 应作为 .05。用户输入的某个利率，如 5 或 7.5 需在应用程序中除以 100，以取得利率计算中所要求的 .05 或 .075。）

我们将以这个完成后的应用程序的探试作为开始。之后，学习另外一些 Java 技术以最终创建一个属于自己的应用程序。



探试利息计算器应用程序

1. 定位到完成后的应用程序 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\Examples\Tutorial10\CompletedApplication\InterestCalculator` 将目录改变到这个完成后的利息计算器应用程序的目录下面。
2. 运行利息计算器应用程序 在命令提示符窗口下键入 `java InterestCalculator` 运行该应用程序(如图 10.1 所示)。用户利用 `JTextField` 输入投资的本金以及年利率的大小。利用 `JSpinner` 组件输入投资的总年数。`JSpinner` 组件可将用户的选择限制在一个特定的范围内——此时，为 1~10 之间的数(包含 1 和 10)。初始值为 1，用户通过点击上下箭头来增加或降低 `JSpinner` 内的值。试着点击相应的上下箭头，对这项功能进行测试。一旦用户输入本金、利率及总年数后，点击 `Calculate JButton`，这时，每年的存款额将显示在应用程序 GUI 底部的 `JTextArea` 内。

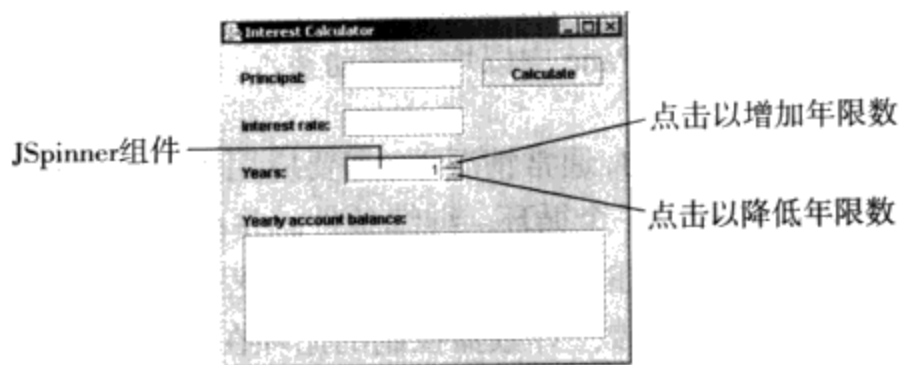


图 10.1 运行完成后的利息计算器应用程序

3. 输入本金 在应用程序运行起来以后，向 `Principal:JTextField` 中输入一个值。如输入应用程序需求分析中所指定的 1000。
4. 输入利率 接下来，还需要在 `Interest rate:JTextField` 中输入一个值。问题陈述中所要求的利率为 5%，因此，在 `Interest rate:JTextField` 中输入 5。
5. 选择存款期限 现在，为该账户下存款额的计算选择希望年限。本例中，通过重复点击 `Years:JSpinner` 中的向上箭头，一直读到 10 为止。
6. 计算资金总额 点击 `Calculate JButton`。该账户下，整个 10 年期期限中每一年年末的资金都将显示在带有滚动条的一个 `JTextArea` 中(参见图 10.2)，此时，用户便可看到所有输出的数据。注意观察滚动条将自动滚动至 `JTextArea` 文本的末尾。通过鼠标向上拖动滚动条，便可以看到最上方所显示的数据了(参见图 10.3)。
7. 再度考察 `JSpinner` 重复点击向下箭头直到 `JSpinner` 中的值为 1。之后，再次点击向下箭头。注意观察 `JSpinner` 的值并没有发生改变。接着，重复点击向上箭头直到 `JSpinner` 中的值为 10。之后，再次点击向上箭头，同样观察到 `JSpinner` 的值也没有发生改变。如果 `JSpinner` 中显示的是某个范围中的最小值(本例中为 1)，点击向下箭头将不会对 `JSpinner` 中的值产生影响。同样，如果 `JSpinner` 中显示的是某个范围中的最大值(本例中为 10)，点击向上箭头也不会对 `JSpinner` 中的值产生影响。还可以利用键盘直接向 `JSpinner` 中的文本域输入一个有效数据(本例中，为一个 0~10 之间的整数)，然后再点击 `Calculate JButton`(参见图 10.4)。注意观察，应用程序同样也能够正常工作。
假若使用键盘输入了一个无效的数据(对于本例，可以是一个不在 0~10 之间的数字，或者是一个如 "hello" 的字符串)，然后再试着点击向上或向下箭头(参见图 10.5 和图 10.6)。注意，当输入无效数据以后，点击向上或向下箭头则会发出蜂鸣声，但不会修改 `JSpinner` 中的值。这时，若点击 `Calculate JButton`，则 `JSpinner` 中的值将显示为最近一次所输入的有效值，然后应用程序将根据这个有效值计算出正确的结果(例如，按照图 10.5 和图 10.6 中的输入，当点击 `Calculate JButton` 时，会得到如图 10.4 的输出结果，而 `JSpinner` 中的值也相应地进行了复位的操作)。在 `JSpinner` 失去焦点后(如点击另一个 GUI 组件)，将会对无效的数据进行相应的复位操作。

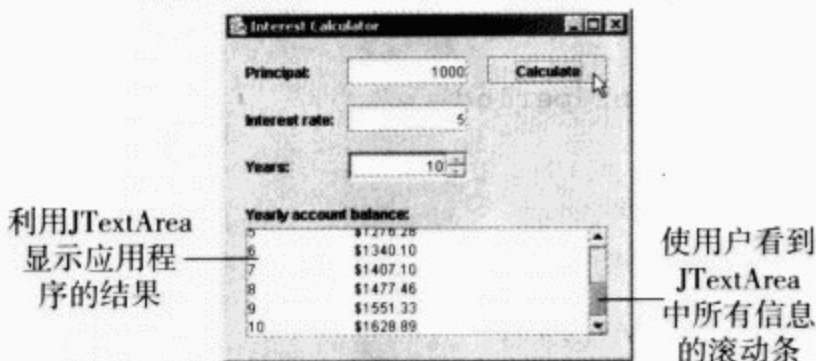


图 10.2 完成后的利息计算器应用程序（最初结果）

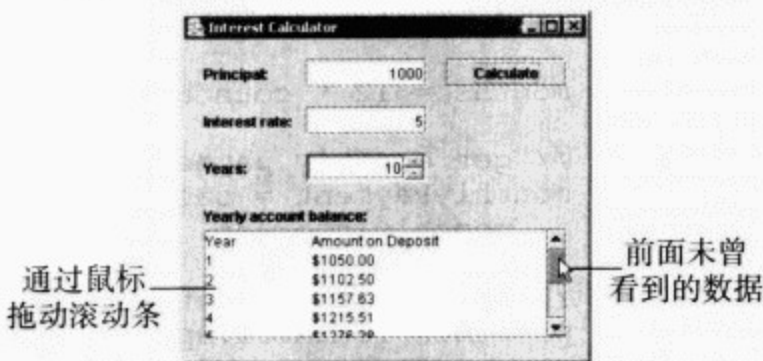


图 10.3 完成后的利息计算器应用程序（向上滚动后的结果）



GUI 设计提示

当想去使用 JTextField 但又希望用户在一个指定的范围内输入有效的数据时，可考虑使用 JSpinner。JSpinner 通过拒绝错误的类型或者是超出某个范围的值来防止无效输入。当无效数据被拒绝以后，JSpinner 中以前曾显示的值将得到恢复。



错误预防提示

利用 JSpinner 可确保只有正确的、处于范围之内数据得到录入。



图 10.4 有效输入时的输出结果



图 10.5 向 JSpinner 中输入超出范围的数据



图 10.6 向 JSpinner 中输入错误类型的数据

8. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。

9. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

10.2 计数器控制循环的要素

本节将通过使用教程8中介绍的while循环语句，找出用于完成计数器控制循环所需要的元素。计数器控制的循环有四个关键要素。它们是：

1. 确定循环是否继续迭代的控制变量（或循环变量）的名称。
2. 控制变量的初始值。
3. 每次循环迭代（即执行每次循环）过程中用于修改控制变量的自增（自减）运算。
4. 测试控制变量最终值的条件（以确定循环是否继续）。

图 10.7 中有一个曾在教程 8 中所完成的应用程序里使用过的 while 循环语句。该 while 语句的计数器由变量 years 表示，为简单起见，我们将图中的这一变量称为 counter。

```
1 int counter = 2 ; // repetition counter
2
```

```
3 while ( counter <= 5 )
4 {
5     months = 12 * counter; // calculate payment period
6
7     // get monthly payment
8     monthlyPayment = calculateMonthlyPayment(
9         monthlyInterest, months, loanAmount );
10
11    // insert result into paymentsJTextArea
12    paymentsJTextArea.append( "\n" + months + "\t" +
13        currency.format( monthlyPayment ) );
14
15    counter++; // increment counter
16
17 } // end while
```

图 10.7 计数器控制循环举例

在本例中，使用到了计数器控制循环中的四个要素。回想汽车贷款计算器应用程序，该程序用于计算并显示贷款期限从 2 年到 5 年时间的汽车月度支付额。第 1 行中的声明是对控制变量（`counter`）进行命名，并指定它为一个 `int` 类型的数据。此声明中同时还包含了一个初始化的过程，即将该变量的值初始值设置为 2。

考察 `while` 语句（第 3 行至第 17 行）。第 5 行通过使用 `counter` 变量，计算出需还款的总月数。第 8 行至第 9 行，调用了教程 8 中已提供给应用程序中的一个用于确定该汽车月度支付额的 `calculateMonthlyPayment` 方法。此方法需要有利率、以月为单位的支付期限，以及汽车的价格这些参数。第 12 行至第 13 行是在 `JTextArea` 中显示数值。第 15 行是对该循环中每次用于迭代的控制变量加 1。`while` 语句中的条件（参见第 3 行）将测试该控制变量的值是否小于或等于 5，这意味着，5 是条件为 `true` 的最终值。`while` 语句体在控制变量为 5 时，同样也会执行。当控制变量超过 5 时（也就是说，当 `counter` 的值为 6 时），循环才会终止执行。

自测题

- 控制变量的 _____ 是计数器控制循环中四个关键要素之一。
a) 最终值 b) 初始值 c) 自增（或自减）运算 d) 以上所有答案
- 计数器控制循环中的哪一个要素将用于确定循环变量在每次的循环迭代中如何进行修改？
a) 名称 b) 初始值 c) 自增（或自减）运算 d) 最终值

答案：1) d 2) c

10.3 引入 for 循环语句

利用 `for` 循环语句，开发人员能够很容易地编写出有关计数器控制循环的执行代码。该语句将方便地指出计数器控制循环中的四个关键要素。为了加强对这种新的循环语句的理解，通过以下内容，将学习如何将图 10.7 中的 `while` 语句替代成一个等效的 `for` 语句。转换后的代码如图 10.8 所示。

```
1 for ( int counter = 2 ; counter <= 5 ; counter += 1 )
2 {
3     months = 12 * counter; // calculate payment period
4
5     // get monthly payment
6     monthlyPayment = calculateMonthlyPayment(
7         monthlyInterest, months, loanAmount );
```



```
8
9 // insert result into paymentsJTextArea
10 paymentsJTextArea.append( "\n" + months + "\t" +
11     currency.format( monthlyPayment ) );
12
13 } // end for
```

图 10.8 展示汽车贷款计算器应用程序中 for 语句的代码片段



好的编程习惯

在控制语句上下位置处留出垂直距离及对控制语句内部进行的缩进处理,都将有助于改善代码的可读性。

让我们一起来看一下这个 for 循环语句是怎样实现的。for 语句的第 1 行(包括关键字 for 及 for 之后圆括号内的所有东西),即图 10.8 中的第 1 行,被非正式地称为 for 语句首部,或简单地叫做 for 首部。for 首部指明了计数器控制循环中的所有四个关键要素。在 for 语句中的第一个表达式,指出了控制变量 counter 的名称和初始值。第二个表达式,则指出了循环控制的条件(这里为 counter <= 5)。最后一个表达式,用于指明相应的自增运算(作为修改每次 for 语句执行过程时 counter 的数值)。此行可读为“对于起始于 2 终止于 5 的每一个 counter 值,完成下列语句,之后对 counter 加 1”。

counter 的初始值为 2,因而满足循环-控制条件,for 语句体内的贷款计算将得到执行。在执行完 for 语句体以后,应到达右花括号(参见第 13 行),即利用其划定 for 语句的结束。当到达右花括号以后,控制又将再次返回到位于第 1 行的 for 首部的最右部分,即自增 counter 并在再次通过循环-控制条件测试的基础之上,重新开始循环。for 语句将重复执行,直到循环-继续条件变为 false(当 counter 大于 5),那时,循环将终止执行。



错误预防提示

尽管控制变量值可在 for 循环的语句体内得到改变,但应避免这种做法,因为这样做有可能会产生某些隐藏的错误出现。

让我们通过图 10.9 更近距离地来看一看图 10.8 中的 for 语句。可以看到,for 语句“包揽了一切”——它利用一个控制变量指明了计数器控制循环中所需要的每一个项目。

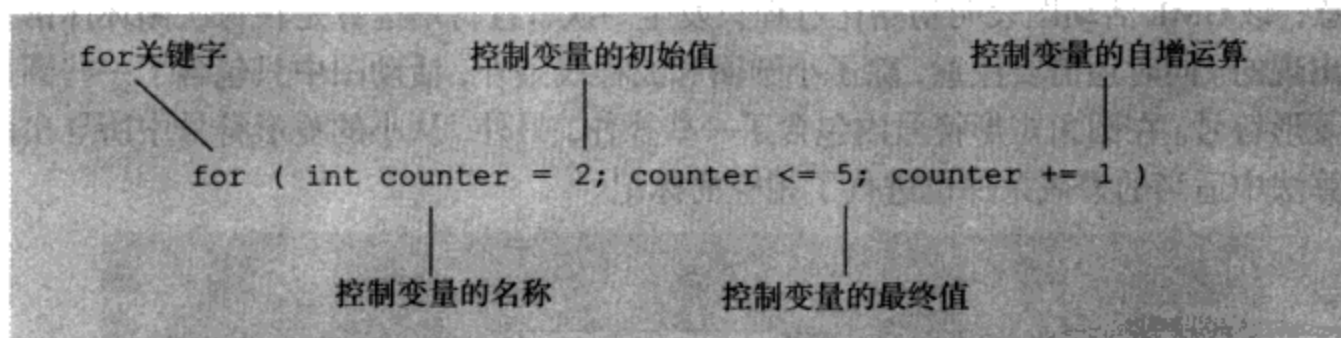


图 10.9 for 首部中的各个组件



错误预防提示

使用 for 循环作为计数器控制的循环,将有助于消除偏一错误(发生在当循环执行的迭代次数比所希望的次数多一次或少一次时),这是因为控制变量的最终值是很清楚的。

每一个 for 语句都是从关键字 for 开始的。然后,该语句将命名并初始化一个控制变量(这里,counter 被设置为 2)[注意:不需要在 for 语句之前声明(利用 int 关键字) counter 变量。这完全可在首部中完成,如图 10.9 所示。如果 counter 变量是按照这种方式进行声明的,则它只可能在 for 语句体的范围内有效。如果 counter 变量还需要在 for 语句之后进行访问,则需要把它放在 for 语句体的外面来声明]。紧跟控制变量初始值之后的是一个分号,再后来是循环-控制的条件。其后也有一个分号,最后便是该控制变量的自增运算。以下框图将用于描述上述循环语句(如图 10.9 所示)的每一步执行过程。

执行 for 循环语句

1. 应用程序声明 int 型变量 counter 并将其初始化为 2。
2. 检查循环-继续条件。循环-继续条件的计算结果为 true (counter 为 2, 小于等于 5), 应用程序开始执行 for 语句体内的语句。
3. 将 counter 的值加 1; counter 中现在包含的值为 3。
4. 检查循环-继续条件。循环-继续条件的计算结果为 true (counter 为 3, 小于等于 5), 应用程序开始执行 for 语句体内的语句。
5. 将 counter 的值加 1; counter 中现在包含的值为 4。
6. 检查循环-继续条件。循环-继续条件的计算结果为 true (counter 为 4, 小于等于 5), 应用程序开始执行 for 语句体内的语句。
7. 将 counter 的值加 1; counter 中现在包含的值为 5。
8. 检查循环-继续条件。循环-继续条件的计算结果为 true (counter 为 5, 小于等于 5), 应用程序开始执行 for 语句体内的语句。
9. 将 counter 的值加 1; counter 中现在包含的值为 6。
10. 检查循环-继续条件。循环-继续条件的计算结果为 false (counter 为 6, 不小于等于 5), 应用程序退出 for 循环。

for 语句中的初始值、结束值及自增运算部分, 均可包含在算术表达式之中。例如, 若 $a = 2$, $b = 10$, 则首部:

```
for ( int i = a; i <= 4 * a * b; i += ( b / a ) )
```

将等价于:

```
for ( int i = 2; i <= 80; i += 5 )
```

如果循环-继续条件的初始值为 false (例如, 若控制变量 i 的初始值为 5, 且循环-继续条件为 $i \leq 4$), 则 for 语句体将不予执行。相反, 执行将转入位于 for 语句体之后的第 1 条语句上。

控制变量经常需要在 for 循环体内的计算中进行显示或者是被使用, 但这并不是一定的。通常, 使用控制变量只是用来控制循环, 而并非一定要在循环中使用它。

图 10.10 中包含了图 10.8 中相关 for 语句的一个 UML 活动图。此活动图与 while 语句的活动图类似。注意, 该 UML 活动图表明初始化过程只发生一次, 且自增运算是在每次循环体语句执行结束之后才出现的。同时还需要注意, 除了小圆圈和流程线条外, 活动图中只包含了一些圆角矩形符号及小的菱形符号。在圆角矩形符号内包含了一些操作, 另外, 从小的菱形符号中所导出的流程线条, 通过算法中适当的警戒条件也进行了相应的标记。

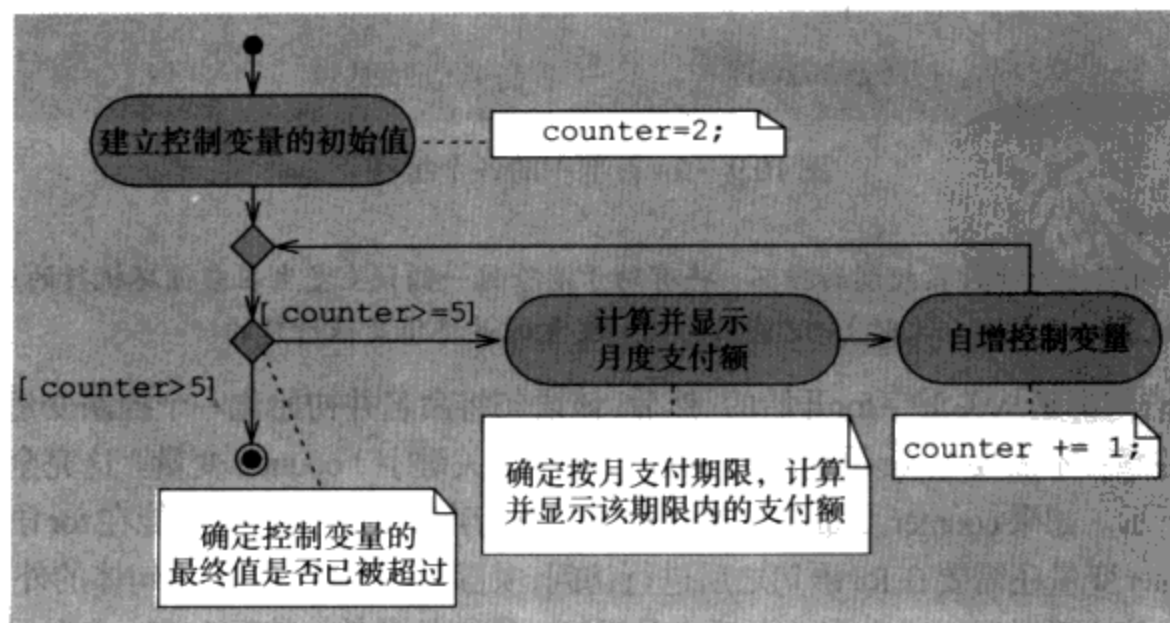


图 10.10 for 循环语句的 UML 活动图

**常见编程错误**

计数器控制的循环不应使用浮点变量进行控制。浮点变量在计算机内存中表示的只是一个近似结果，因而，有可能会产生不精确的计数值以及因终止条件的不准确性测试而出现逻辑错误。

**常见编程错误**

使用一个无法修改控制变量值的表达式（如将 `counter++` 和 `counter+=1` 错误地写成了 `counter+1`），往往会导致出现一个无限循环。

自测题

1. for 语句内，位于第一个分号前的值表示 ____。
a) 计数器变量的初始值 b) 计数器变量的最终值 c) 自增运算 d) 语句迭代的次数
2. for 循环语句的第 1 行常被非正式地称为 ____。
a) for 语句体 b) for 首部 c) for 计数器 d) for 表达式

答案：1) a 2) b

10.4 for 循环举例

在以下例子中，将介绍如何根据各种不同的方式来改变 for 语句中的控制变量。对于每一种情况，都将提供适当的 for 首部。

- 按照自增 1 的方式将控制变量从 1 改变到 100。

```
for( int i = 1; i <= 100; i++ )
```

- 按照自增 -1 的方式将控制变量从 100 改变到 1（即自减 1）。注意，现在进行的是向下计数，循环 - 继续条件为 $i \geq 1$ ，其中，使用到了关系运算符 \geq 。

```
for( int i = 100; i >= 1; i-- )
```

- 按照自增 7 的方式将控制变量从 7 改变到 77。

```
for( int i = 7; i <= 77; i += 7 )
```

- 按照自增 -2 的方式将控制变量从 20 改变到 2（即自减 2）。

```
for( int i = 20; i >= 2; i -= 2 )
```

- 按照下列值的顺序：2, 5, 8, 11, 14, 17, 20，改变控制变量。

```
for( int i = 2; i <= 20; i += 3 )
```

- 按照下列值的顺序：99, 88, 77, 66, 55, 44, 33, 22, 11, 0，改变控制变量。

```
for( int i = 99; i >= 0; i -= 11 )
```

自测题

1. 下面哪一个 for 首部能够按照以下值的顺序：25, 20, 15, 10, 5，正确地改变其中的控制变量？
a) `for(int i = 5; i <= 25; i += 5)` b) `for(int i = 25; i >= 5; i -= 5)`
c) `for(int i = 5; i <= 25; i -= 5)` d) `for(int i = 25; i >= 5; i += 5)`
2. 下面哪句话能够正确地描述下列 for 语句的首部？

```
for ( int i = 81; i <= 102; i ++ )
```

- a) 按照自增 1 的方式将控制变量从 81 改变到 102
- b) 按照自增 0 的方式将控制变量从 81 改变到 102

- c) 按照自增 -1 的方式将控制变量从 102 改变到 81
- d) 按照自增 2 的方式将控制变量从 81 改变到 102

答案: 1) b 2) a

10.5 创建利息计算器应用程序

下面将创建自己的利息计算器应用程序, 需要使用到一个管理输入的 JSpinner, 用于显示大量数据的滚动条以及一条计算每年年末投资额价值的 for 语句。以下的这个伪代码描述了该利息计算器应用程序的基本操作, 它们是在用户点击 Calculate JButton 时开始执行的。

当用户点击 Calculate JButton 时
 获取用户输入的本金、利率及年限长度
 在 JTextArea 中显示一个用于标记输出的表头

 对于每一年, 起始于 1 结束于用户输入的年限长度
 计算并显示该年
 计算并显示出投资额的当前价值

在本教程中使用的这个模板应用程序将包含: Calculate JButton, 两个 JLabel 及与它们对应的 JTextField——Principal:JLabel 与对应的 JTextField 和 Interest rate:JLabel 与对应的 JTextField。应用程序中还有一个 Years:JLabel, 但读者需为这一输入自定义 JSpinner 组件。还需要自定义一个带滚动条 JTextArea, 并把它加入到应用程序中的 GUI 内。最后, 将使用一条 for 语句为应用程序添加相应的功能。

我们已对利息计算器应用程序进行了探试并研究了它的伪代码表示, 下面, 将使用一张用于帮助读者把这个伪代码转换成 Java 实现的 ACE 表。图 10.11 中列出了该应用程序中相应的操作、组件以及事件, 以帮助读者最终完成属于自己版本的这一应用程序。


操作	组件	事件
 标记应用程序中的组件	interestRateJLabel	当应用程序运行时
	principalJLabel	
获取用户输入的本金、利率及年限长度	yearsJLabel	当用户点击 Calculate JButton 时
	yearlyBalanceJLabel	
	calculateJButton	
	principalJTextField	
	interestRateJTextField	
在 JTextArea 中显示一个用于标记输出的表头	yearsJSpinner	当用户点击 Calculate JButton 时
	yearlyBalanceJTextArea	
	yearlyBalanceJTextArea	
对于每一年, 起始于 1 结束于用户输入的年限长度	yearlyBalanceJTextArea	当用户点击 Calculate JButton 时
计算并显示该年	yearlyBalanceJTextArea	
计算并显示出投资额的当前价值	yearlyBalanceJTextArea	

图 10.11 利息计算器应用程序的 ACE 表

下面, 将开始创建自己的利息计算器应用程序。首先, 需自定义一个 JSpinner 组件, 允许用户在某个指定的范围内利用 JSpinner 的向上箭头和向下箭头选择一个年限长度。接着, 学习如何设置 JSpinner 的范围界限 (最大和最小值)。我们将使用 1 作为该组件的最小值, 使用 10 作为它的最大值。

自定义一个 JSpinner 组件

1. 将模板复制到工作目录中 将 C:\Examples\Tutorial10\TemplateApplication\InterestCalculator 目录复制到 C:\SimplyJava 目录中。
2. 打开班级平均分应用程序的模板文件 在自己的文本编辑器中打开模板文件 InterestCalculator.java。

3. 设置 JSpinner 的范围值 将图 10.12 中第 83 行上位于 SpinnerNumberModel 之后两个圆括号之间的值 (该值已被突出显示), 输入到模板代码中。SpinnerNumberModel 为一个类, 它指明该 JSpinner 中将包含一个数值范围而不是其他的数据类型, 比如日期等。将在教程 25 中学习如何利用 JSpinner 显示日期。在第 82 行至第 83 行上的语句, 指明了包含在这个 JSpinner 范围中的一些值。第一个值为 1, 用于指明当应用程序运行时, JSpinner 中所显示的初始值。下面的两个值 1 和 10, 分别指出了该范围内的最大值和最小值。最后一个值 (或称步长) 为 1, 表示该范围中的每一个值将比前一个值大 1。所谓步长是指, 当用户点击当前 JSpinner 组件的向上箭头或向下箭头时, 其数值的变化幅度。



好的编程习惯

在 JSpinner 组件名的后面添加单词 JSpinner 作为后缀。

创建 JSpinner
上的数据并
设置其范围

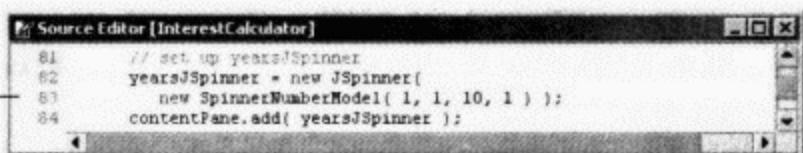


图 10.12 指定 yearsJSpinner 的初始值 (为 1)、范围 (为 1~10) 及步长值 (为 1)

4. 设置 JSpinner 组件的范围 添加图 10.13 中的第 84 行, 将 yearsJSpinner 的 bounds 属性设置为 100, 96, 100, 24, 使之能够同上面的一些 JTextField 对齐。

设置 yearsJSpinner
的 bounds 属性

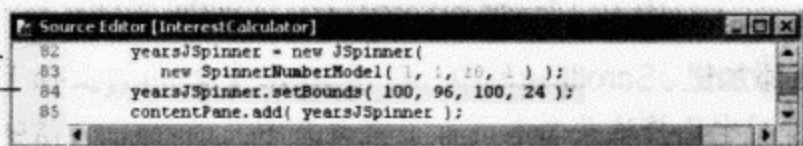
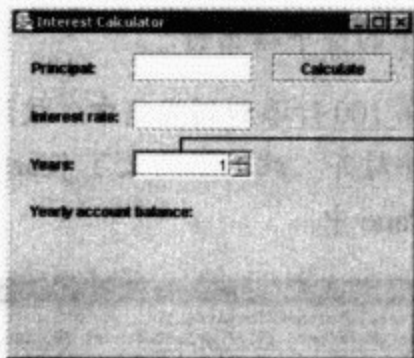


图 10.13 设置 JSpinner 的 bounds 属性

5. 保存应用程序 保存修改后的源代码文件。
6. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\InterestCalculator` 进入到当前的工作目录中。
7. 编译应用程序 通过键入 `javac InterestCalculator.java` 编译该应用程序。
8. 运行应用程序 若此应用程序能够正确编译, 通过键入 `java InterestCalculator` 来运行它。图 10.14 中显示了更新后的应用程序的运行结果。注意观察, 该 JSpinner 中包含的初始值为 1。利用上下箭头可改变其中所显示的值。另外, 注意当点击 Calculate JButton 时, 并无任何操作产生。将在本教程随后的部分中添加进这一功能。



JSpinner 组件

图 10.14 添加到利息计算器应用程序中的 JSpinner

9. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
10. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。



GUI 设计提示

为 JSpinner 和 JTextField 使用相同的 GUI 设计原则 (参见附录 C, GUI 设计原则)。

利息计算器应用程序通过一个 JTextArea 显示出了相应的计算结果。需要为这个 JTextArea 配置一个滚动条,这样,当 JTextArea 太小而不能显示更多的内容时,用户可通过上下滚动的方式查看到该框内的全部内容。为完成这项工作,Java 提供了一个 JScrollPane 组件。JScrollPane 是一个容器,这意味着其可包含其他的组件(类似于应用程序的内容面板)。JScrollPane 并不是一个直接可以看到的组件,但是,它却能够为某个添加到其内部的组件在必要的时候显现出滚动条。反过来,JScrollPane 自身又需要被添加到内容面板中。存储在 JScrollPane 中的组件,在平时其外观是一样的,仅仅除了在必要的时候才会显示出相应的滚动条,这实际取决于所显示的数据量。接下来,通过使用一个 JScrollPane 将滚动条添加到 JTextArea 中。

自定义一个拥有滚动条的 JTextArea

1. 设置可编辑属性 将图 10.15 中第 95 行添加到代码中。此行将把 editable 属性设置为 false,以使用户不能在 Yearly account balance:JTextArea 中改变输出。

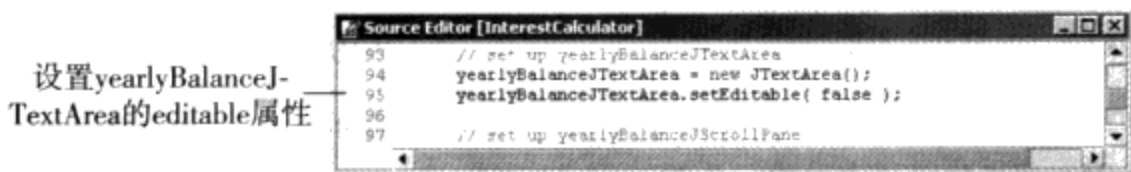


图 10.15 将 JTextArea 的 editable 属性设置为 false

2. 将 JTextArea 添加进 JScrollPane 中 将图 10.16 中第 99 行上位于 JScrollPane 后圆括号之间的内容添加到代码中(图中已将这文本进行了突出显示)。注意,为清晰起见,我们将这条语句的一部分放置到了下一行上。此行把 JTextArea 同 JScrollPane 联系了起来,使 JScrollPane 在必要时,向 JTextArea 中加入滚动条,即若有更多文本需放置在 JTextArea 中所能容纳的范围宽度时,会出现一个水平滚动条;而若有更多文本需放置在 JTextArea 中所能容纳的范围高度时,则会出现一个垂直滚动条。

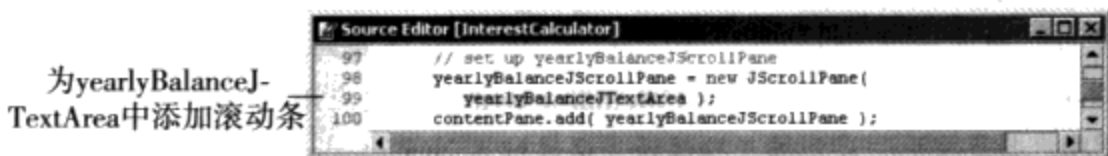


图 10.16 将 yearlyBalanceJTextArea 添加到 yearlyBalanceJScrollPane 中



GUI 设计提示

若某个 JTextArea 中需要按照多行的形式显示输出,可将这一 JTextArea 同 JScrollPane 联系起来,以使用户通过滚动的方式看到 JTextArea 中的整个输出行。

3. 设置 JScrollPane 的范围 将图 10.17 中第 100 行添加到代码中。该语句将 JScrollPane 的 bounds 属性设置为 16, 160, 300, 92, 以便同上面的组件对齐。此范围定义了 yearlyBalanceJTextArea 的大小和位置,其本身也将包含在 yearlyBalanceJScrollPane 中。

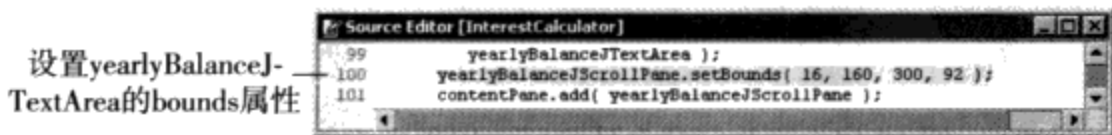


图 10.17 添加一个可容纳 JTextArea 的 JScrollPane

4. 保存应用程序 保存修改后的源代码文件。
5. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\InterestCalculator` 进入到当前的工作目录中。
6. 编译应用程序 通过键入 `javac InterestCalculator.java` 编译该应用程序。

- 5. 编译应用程序 通过键入 javac InterestCalculator.java 编译该应用程序。
- 6. 运行应用程序 若此应用程序能够正确编译，通过键入 java InterestCalculator 来运行它。图 10.21 中显示了更新后的应用程序的运行结果。输入本金、利率及年限数，然后按下 Calculate JButton。可以看到，表头出现在了 JTextArea 中。将在下一个框图中，添加进用以产生出其他输出结果的功能。注意观察，滚动条仍然没有出现，这是因为此时的输出结果并未充满整个 JTextArea。

在JTextArea内
放置一个表头
并创建一个用于
格式化美元值的
DecimalFormat



图 10.20 在 JTextArea 内显示一个表头的应用程序代码

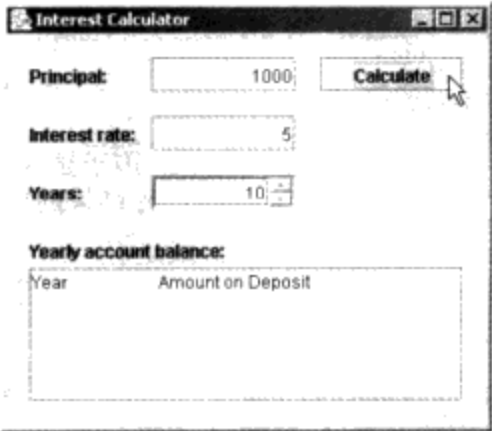


图 10.21 在 JTextArea 中输出的表头

- 7. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- 8. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

接下来，我们将写for语句来完成指定年限的计算，还将学习用Math.pow方法来完成求幂运算。

利用一条 for 语句创建累积利息

- 1. 创建一条空的 for 语句 将图 10.22 中第 145 行至第 149 行插入到 calculateJButtonActionPerformed 事件处理程序中。第 146 行的 for 首部将把控制变量初始化为 1。位于第一个分号后的测试（循环-继续条件）会在控制变量小于或等于由用户所指定的年限数时使循环继续执行。而 for 语句在该年限数到达 year 值的过程中反复执行其语句体，另外，控制变量 count 也将按照自增 1 的方式从 1 变化到 year 值。

空的for语句

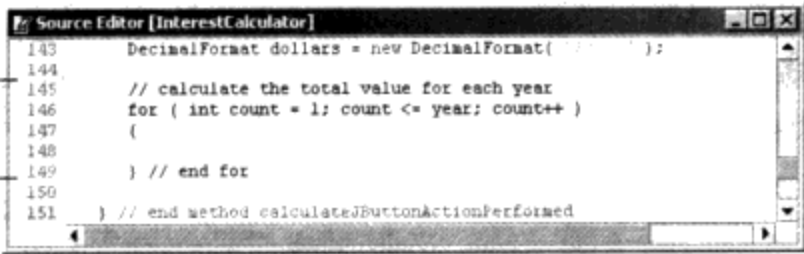


图 10.22 创建 for 语句

- 2. 完成利息计算 将图 10.23 中第 148 行至第 149 号插入到 for 语句体内。第 148 行至第 149 行利用下面的公式完成了相应的计算：

$$a = p (1 + r)^n$$

其中，a 为总数额，p 为本金，r 为利率，n 为年限数。

第 149 行通过调用 Math.pow 方法完成了乘幂的运算。Math.pow 有两个参数——第一个参数为需进行乘幂运算的某个数，第二个参数为需进行乘幂运算的第一个参数的乘幂值。例如，Math.pow(4, 2)代表表达式 4²，其计算的结果为 16。

通过Math的pow
方法计算指定年
限数的存款数额

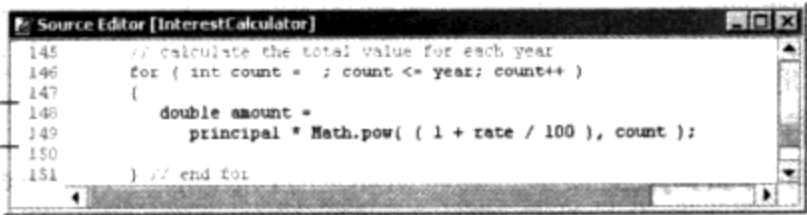


图 10.23 计算总数额的应用程序代码

第 149 行使用的是公式中的 $(1+r)^n$ 部分。Math.pow 的第一个参数为 $(1 + \text{rate}/100)$ ，代表公式中的 $(1+r)$ 部分。除以 100 表示将输入（应为百分比格式）转化成一个可在利息计算中所使用的小数值。Math.pow 的第二个参数 counter，表示对 $(1 + \text{rate}/100)$ 执行的乘幂（即上述公式中的 n ）运算。

3. 将计算结果追加到 JTextArea 中 将图 10.24 中的第 150 行至第 151 行添加到 for 语句中。这几行利用 append 方法，将额外的文本追加到了输出 JTextArea 的末尾位置处。该文本中将包括：用于在下一行中进行输出的换行符（“\n”）、当前的 count 值、用于定位至下一列的 tab 字符（“\t”），还有调用方法 dollars.format(amount) 以返回货币格式的值 amount。

循环体执行完毕以后，应用程序将到达位于第 153 行上的右花括号处。计数器（count）将自增 1，然后，循环又继续开始了对 for 首部中循环-继续条件的测试。for 语句将继续执行直到控制变量超过用户所指定的年限值。

4. 保存应用程序 保存修改后的源代码文件。

将一行文本追加到 JTextArea 中
(注意，利用转移序列\n，可生成一行新的文本，而转移序列\t，则使文本定位到第二列上。)

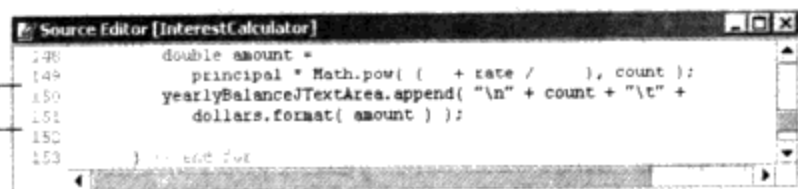


图 10.24 将输出附加到 yearlyBalanceJTextArea 中

5. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\InterestCalculator 进入到当前的工作目录中。
6. 编译应用程序 通过键入 javac InterestCalculator.java 编译该应用程序。
7. 运行应用程序 若此应用程序能够正确编译，通过键入 java InterestCalculator 来运行它。图 10.25 中显示了完成后的应用程序的运行结果。注意，此时的输出要比 JTextArea 中所能容纳的范围更大，因而会出现了一个垂直滚动条。利用这个垂直滚动条便可考察到整个的输出结果（如图 10.26 所示）。



图 10.25 完成后的应用程序（最初结果）

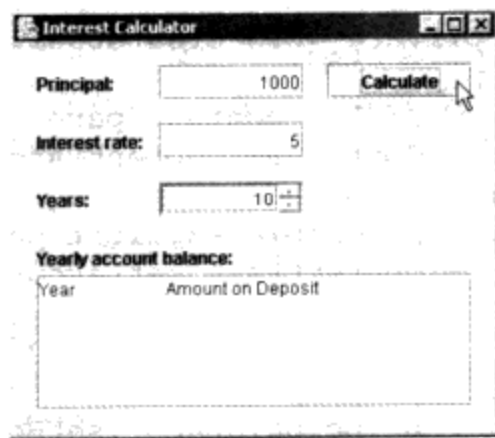


图 10.26 完成后的应用程序（其余结果）

8. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
9. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

图 10.27 中给出了利息计算器应用程序的完整源代码。本教程中，凡需要添加、查看或是修改的代码，均在图中相应的代码行中做了突出显示。

```

1 // Tutorial 10: InterestCalculator.java
2 // Calculate the total value of an investment.
3 import java.awt.*;
4 import java.awt.event.*;
5 import javax.swing.*;
6 import java.text.*;
7
8 public class InterestCalculator extends JFrame

```

```
9 {
10     // JLabel and JTextField for principal invested
11     private JLabel principalJLabel;
12     private JTextField principalJTextField;
13
14     // JLabel and JTextField for interest rate
15     private JLabel interestRateJLabel;
16     private JTextField interestRateJTextField;
17
18     // JLabel and JTextField for the number of years
19     private JLabel yearsJLabel;
20     private JSpinner yearsJSpinner;
21
22     // JLabel and JTextArea display amount on deposit at
23     // the end of each year up to number of years entered
24     private JLabel yearlyBalanceJLabel;
25     private JTextArea yearlyBalanceJTextArea;
26
27     // JScrollPane adds scrollbars to JTextArea for lengthy output
28     private JScrollPane yearlyBalanceJScrollPane;
29
30     // JButton calculates amount on deposit at the
31     // end of each year up to number of years entered
32     private JButton calculateJButton;
33
34     // no-argument constructor
35     public InterestCalculator()
36     {
37         createUserInterface();
38     }
39
40     // create and position GUI components; register event handlers
41     private void createUserInterface()
42     {
43         // get content pane for attaching GUI components
44         Container contentPane = getContentPane();
45
46         // enable explicit positioning of GUI components
47         contentPane.setLayout( null );
48
49         // set up principalJLabel
50         principalJLabel = new JLabel();
51         principalJLabel.setBounds( 16 , 16 , 56 , 24 );
52         principalJLabel.setText( "Principal:" );
53         contentPane.add( principalJLabel );
54
55         // set up principalJTextField
56         principalJTextField = new JTextField();
57         principalJTextField.setBounds( 100, 16 , 100, 24 );
58         principalJTextField.setHorizontalAlignment(
59             JTextField.RIGHT );
60         contentPane.add( principalJTextField );
61
62         // set up interestRateJLabel
63         interestRateJLabel = new JLabel();
64         interestRateJLabel.setBounds( 16 , 56 , 80 , 24 );
65         interestRateJLabel.setText( "Interest rate:" );
66         contentPane.add( interestRateJLabel );
67
68         // set up interestRateJTextField
69         interestRateJTextField = new JTextField();
```



```

70     interestRateJTextField.setBounds( 100, 56 , 100, 24 );
71     interestRateJTextField.setHorizontalAlignment(
72         JTextField.RIGHT );
73     contentPane.add( interestRateJTextField );
74
75     // set up yearsJLabel
76     yearsJLabel = new JLabel();
77     yearsJLabel.setBounds( 16 , 96 , 48, 24 );
78     yearsJLabel.setText( "Years:" );
79     contentPane.add( yearsJLabel );
80
81     // set up yearsJSpinner
82     yearsJSpinner = new JSpinner(
83         new SpinnerNumberModel( 1, 1, 10, 1 ) ); // 设置JSpinner的范围
84     yearsJSpinner.setBounds( 100, 96 , 100, 24 ); // 自定义yearsJSpinner的bounds属性
85     contentPane.add( yearsJSpinner );
86
87     // set up yearlyBalanceJLabel
88     yearlyBalanceJLabel = new JLabel();
89     yearlyBalanceJLabel.setBounds( 16 , 136, 150 , 24 );
90     yearlyBalanceJLabel.setText( "Yearly account balance:" );
91     contentPane.add( yearlyBalanceJLabel );
92
93     // set up yearlyBalanceJTextArea // 将 yearlyBalanceJTextArea
94     yearlyBalanceJTextArea = new JTextArea(); // 的editable属性设置为false
95     yearlyBalanceJTextArea.setEditable( false );
96
97     // set up yearlyBalanceJScrollPane
98     yearlyBalanceJScrollPane = new JScrollPane(
99         yearlyBalanceJTextArea ); // 将一个JTextArea添加到JScrollPane中并设置该
100    yearlyBalanceJScrollPane.setBounds( 16 , 160, 300, 92 ); // JScrollPane的范围
101    contentPane.add( yearlyBalanceJScrollPane );
102
103    // set up calculateJButton
104    calculateJButton = new JButton();
105    calculateJButton.setBounds( 216 , 16 , 100, 24 );
106    calculateJButton.setText( "Calculate" );
107    contentPane.add( calculateJButton );
108    calculateJButton.addActionListener(
109        new ActionListener() // anonymous inner class
110        {
111            // event handler called when calculateJButton is clicked
112            public void actionPerformed((ActionEvent event) )
113            {
114                calculateJButtonActionPerformed( event );
115            }
116        } // end anonymous inner class
117    ); // end call to addActionListener
118
119    // set properties of application's window
120    setTitle( "Interest Calculator" ); // set title bar text
121    setSize( 340 , 296 ); // set window size
122    setVisible( true ); // display window
123
124    } // end method createUserInterface
125
126    // calculate and display amounts
127    private void calculateJButtonActionPerformed( ActionEvent event )
128

```

```

132    {
133        // declare variables to store user input
134        double principal = Double.parseDouble(
135            principalJTextField.getText() );
136        double rate = Double.parseDouble(
137            interestRateJTextField.getText() );
138
139        Integer integerObject = ( Integer ) yearsJSpinner.getValue();
140        int year = integerObject.intValue();
141
142        yearlyBalanceJTextArea.setText( "Year\tAmount on Deposit" );
143        DecimalFormat dollars = new DecimalFormat( "$0.00" );
144
145        // calculate the total value for each year
146        for ( int count = 1 ; count <= year; count++ )
147        {
148            double amount =
149                principal * Math.pow( ( 1 + rate / 100 ), count );
150            yearlyBalanceJTextArea.append( "\n" + count + "\t" +
151                dollars.format( amount ) );
152
153        } // end for
154
155    } // end method calculateJButtonActionPerformed
156
157    // main method
158    public static void main( String[] args )
159    {
160        InterestCalculator application = new InterestCalculator();
161        application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
162
163    } // end method main
164
165 } // end class InterestCalculator

```

取得用户输入

为输出显示表头及创建一个用于 格式化输出的对象

使用一条用于计算存款额的for语句,并将相应数据追加到JTextArea中

图 10.27 利息计算器应用程序的完整源代码

自测题

- _____ 可用来确定当用户点击向上箭头或向下箭头时, JSpinner 组件内的数值变化大小。
a) 最小值 b) 步长 c) 最大值 d) value 属性
- _____ 方法可返回一个 Integer 对象的 int 数据。
a) intData b) getValue c) getData d) intValue

答案: 1) b 2) d

10.6 小结

在本教程中, 我们了解了计数器控制循环中的关键要素, 它们是: 控制变量、控制变量的初始值、每次循环中用于修改控制变量的自增 (或自减) 运算以及测试控制变量最终值的条件。接着, 对for循环语句进行了探索, 了解到该循环语句通过一个首部结合了计数器控制循环中的所有要素。

在熟悉了for循环语句以后, 我们看到了如何将汽车贷款计算器应用程序中的 while 语句改成一个for语句。然后, 在分析了相应的伪代码和ACE表之后, 又创建了一个利息计算器应用程序。在利息计算器GUI中, 学到了一些新的设计元素, 其中包括一个JSpinner组件和一个位于JScrollPane内的JTextArea。

在下一个教程中，将学习如何使用 switch 多向选择语句。我们曾经学习过 if...else 选择语句，它是按照某个条件的值在多个操作过程之间进行选择。而将在下一教程中学习的这个 switch 多向选择语句，适合于条件数目巨大时使用，因为，通过它能够节省开发的时间并改善代码的可读性。我们将通过使用一条 switch 多向选择语句，构建出一个安检面板应用程序。

技术小结

使用 for 循环语句

- 在第一个分号前指定控制变量的初始值。
- 在第一个分号后指定循环 - 继续条件。
- 在第二个分号后指定自增（或自减）运算。
- 使用花括号（{和}）确定 for 循环语句体的界限。

执行求幂运算

- 调用 Math.pow 方法。
- 将传递给 Math.pow 方法的第一个参数作为求幂运算的某个数值。
- 将传递给 Math.pow 方法的第二个参数作为该数值的幂值。

从 JSpinner 组件中取得输入

- 调用 getValue 方法。
- 利用 Integer，将返回值转换为一个 Integer 对象。
- 通过使用 intValue 方法从 Integer 对象中取得 int 数据。

关键术语

容器 作为包含其他组件的一种对象。

控制变量 可用于表示计数器控制循环中的迭代次数的一种变量。

控制变量的最终值 在计数器控制循环结束前，控制变量中所存储的一个最后值。

for 关键字 开始于每条 for 语句之前的关键字。

for 语句首部 /for 首部 位于 for 语句的第一行上。for 首部指明了计数器控制循环的所有四个关键要素——控制变量的名称、初始值、自增或自减运算以及最终值。

for 循环语句 一种能很方便地处理计数器控制循环细节的循环语句。for 首部中使用了计数器控制循环的四个关键要素。

JSpinner 类的 getValue 方法 可用于返回 JSpinner 组件中所显示的当前值。

控制变量的自增（或自减） 用以表示每次循环迭代过程中控制变量值的改变幅度。

控制变量的初始值 当计数器控制循环开始时，在控制变量中所存储的值。

Integer 对象 包含了一个 int 数据的一种对象。

Integer 类的 intValue 方法 可返回 Integer 对象中的基本类型 int 数据的一个方法。

JScrollPane 组件 可将滚动条加入到另一个组件中。当某个组件被添加到 JScrollPane 中以后，该组件会在必要时显示出滚动条（即当被添加的组件中所出现的信息比其能显示的信息更多时）。

JSpinner 组件 可将用户的输入限定到一个特定值的范围内。程序员应指出该范围的最大值和最小值、用户点击向上（或下）箭头时的自增（或自减）运算，以及所显示的初始值。

Math.pow 方法 一个用于完成求幂运算的方法。第一个参数指明了需进行求幂运算的数值，第二个参数指明的是针对第一个参数所采取的幂值。

控制变量的名称 可用做访问某个循环控制变量的标识符。

SpinnerNumberModel 对象 用以指明 JSpinner 中所包含数值范围的对象。

步长 指明当用户点击 JSpinner 组件的向上箭头（用于自增）或向下箭头（用于自减）时，JSpinner 中值的改动幅度。

GUI 设计导航

JTextArea

- 若某个 JTextArea 中需要按照多行的形式显示输出, 可将这一 JTextArea 同 JScrollPane 联系起来, 以使用户通过滚动的方式看到 JTextArea 中的整个输出行。

JSpinner

- 当想使用 JTextField 但又希望用户在一个指定的范围内输入有效的数据时, 可考虑使用 JSpinner。JSpinner 通过拒绝错误的类型或者是超出某个范围的值来防止无效输入。当无效数据被拒绝以后, JSpinner 中以前曾显示的值将得到恢复。
- 为 JSpinner 和 JTextField 使用相同的 GUI 设计原则 (参见附录 C)。

Java 类库索引

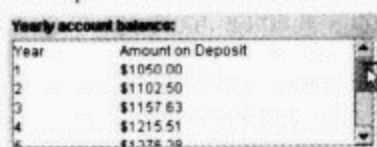
Integer 此类可用于表示及管理 int 型数据。

- 方法

intValue 返回 Integer 对象中的一个 int 数据。

JScrollPane 该组件可向其他的组件中添加水平和/或垂直滚动条。当某个组件, 如 JTextArea, 被添加至一个 JScrollPane 中时, 该组件需在必要时显示出滚动条。

- 运行



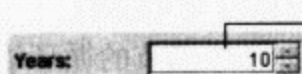
包含在某个 JScrollPane 中的组件, 其在必要时显示出了一个滚动条

- 方法

setBounds 设置 JScrollPane 组件的范围 (位置和大小)。

JSpinner 该组件允许用户在一个特定的范围内指定输入。范围可以是数值, 或者是其他类型的值。其中, 输入可通过键盘键入, 或者是点击 JSpinner 右侧所提供的上下箭头。无效输入将被拒绝并被替换为最近所输入的有效值。

- 运行



JSpinner 组件

- 方法

getValue 返回 JSpinner 组件的 value 属性的数据。

setBounds 设置某个 JSpinner 组件的范围 (位置和大小)。

Math 此类中含有一些常见的算术运算方法。

- 方法

pow 完成求幂运算。第一个参数指明了需进行求幂运算的数值, 第二个参数指明了针对第一个参数所采取的幂值。

习题

选择题

- 10.1 JSpinner 组件允许开发人员指定 _____。
- a) 供用户选择的最大值 b) 供用户选择的最小值
c) 提供给用户的步长 d) 以上答案都正确
- 10.2 _____ 将用于确定循环是否继续迭代。
- a) 控制变量的初始值 b) 右花括号 c) 左花括号 d) 控制变量

- 10.3 在一个 for 循环中, 对控制变量的自增 (或自减) 运算是发生在 ____。
- a) 循环体执行结束以后 b) 循环体执行结束以前
c) 当循环 - 继续条件为 false 时 d) 当执行循环体时
- 10.4 一个 ____ 对象将包含 int 型的数据。
- a) IntegerObject b) Int c) IntData d) Integer
- 10.5 以下哪一个 for 首部 ____ 将按照从 1 到 10 之间的奇数来改变控制变量。
- a) for(int i = 1; i <= 10; i += 1) b) for(int i = 1; i <= 10; i += 2)
c) for(int i = 1; i <= 10; i -= 1) d) for(int i = 1; i <= 10; i -= 2)
- 10.6 拥有首部 for(int i = 0; i <= 50; i += 5) 的 for 循环体将执行 ____ 次。
- a) 50 b) 10 c) 11 d) 以上答案都不对
- 10.7 ____ 方法将返回 JSpinner 内的当前值。
- a) intValue b) getValue c) getNumber d) getCurrentValue
- 10.8 JScrollPane 是一个 ____, 这意味着其他对象可被加入到其内部。
- a) 容器 b) 支架 c) 外壳 d) JFrame
- 10.9 拥有首部 for(int i = 1; i <= 10; i -= 1) 的 for 循环体将执行 ____ 次。
- a) 非常多的数量 b) 9 c) 10 d) 11
- 10.10 Math. ____ 方法可返回对某数求幂后的结果。
- a) power b) exponent c) pow d) exp

练习题

- 10.11 (当前值计算器应用程序) 某银行希望为用户展示将投资多少资金便可在 5, 10, 15, 20, 25 或 30 年期限内获得所预期的财政目标 (未来价值)。用户需提供他们的财政目标 (特定年限后的资金额)、一个利率和投资年限。创建一个这样的应用程序, 计算并显示用于取得用户财政目标所需要的本金 (投资的初始额)。该应用程序允许用户进行投资的年限由 5, 10, 15, 20, 25 或 30 年。比如, 若某个用户希望在 5 年期限达到后 (利率为 5%) 取得 \$15 000 的财政目标, 那么该用户的投资数额需为 \$10 896.96, 其结果如图 10.28 中所示。
- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial10\Exercises\PresentValue 目录复制到 C:\SimplyJava 目录中。
- b) 打开模板文件 在自己的文本编辑器中打开 PresentValue.java 文件。
- c) 自定义 JSpinner 必须自定义显示年限数的 JSpinner。此 JSpinner 的名字为 yearsJSpinner。这个 JSpinner 应按 5 的倍数从 0 显示到 30。修改第 83 行, 使该 JSpinner 的初始值为 0, 最小值为 0, 最大值为 30, 步长为 5。在第 84 行, 插入将其 bounds 属性设置为 130, 95, 100, 20 的代码。
- d) 自定义 JTextArea 必须自定义可显示出各种不同存款额的 JTextArea。此 JTextArea 的名字为 amountNeededJTextArea。在第 94 行插入一个空白行。在第 95 行, 插入可将其 bounds 属性设置为 20, 155, 320, 115 的代码。紧接着下一行, 插入将 amountNeededJTextArea 的 editable 属性设置为 false 的代码。
- e) 从 JSpinner 中获取输入 在第 133 行, 访问 JSpinner 的 value 属性并使用 Integer, 将结果转化为 Integer 类型的一个对象。然后把结果存储到 Integer 的对象 integerObject 中。在第 133 行插入一个空白行。在第 134 行, 取得 integerObject 中的 int 数据, 并将该数据存储到 int 型的变量 years 中。
- f) 完成 for 语句的首部 在第 140 行, 会看到一条只含有两个分号的 for 语句首部。在第一个分号之前, 声明变量 counter 并将其初始化为 5。在第二个分号之前, 插入一个可导致 for 语句不断循环直至 counter 到达用户所指定年限数的循环 - 继续条件。在第二个分号之后, 输入 counter 的自增运算, 使 for 语句能够按照 5 年间隔执行一次。

- g) **计算现值** 下面, 将计算出每 5 年的时间间隔, 所取得的预期价值的数额。为完成这项工作, 需要在 for 语句中实现下面的公式:

$$p = a / (1 + r)^n$$

其中:

p 为达到预期值所需的投资数额

r 为年利率

n 为年限数

a 为预期数额 (用户希望经过 n 年以后的价值所得)

在第 142 行至第 143 行, 利用 Math.pow 方法 (以及在第 129 行和第 131 行已经为读者定义好的变量) 计算当前年限时的现值。使用了两行语句是为了增强代码的清晰性。在第 144 行至第 145 行, 利用 append 方法在应用程序的 JTextArea 中输出计算出的现值。需使用到第 137 行已创建好的 DecimalFormat 对象 (dollars)。同样也使用了两行语句, 是为了增强代码的清晰性。

- h) **保存应用程序** 保存修改后的源代码文件。
- i) **打开命令提示符窗口改变目录** 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\PresentValue 进入到当前的工作目录中。
- j) **编译应用程序** 通过键入 javac PresentValue.java, 编译该应用程序。
- k) **运行完成后的应用程序** 若应用程序能正确编译, 通过键入 java PresentValue 来运行它。输入预期值、利率和年限数, 然后点击 Calculate JButton。查看结果以确保显示出正确的年限数, 并保证初始的存款额也是正确的。
- l) **关闭应用程序** 点击关闭按钮关闭正在运行的应用程序。
- m) **关闭命令提示符窗口** 点击命令提示符窗口的关闭按钮将其关闭。

10.12 (复利: 对比率应用程序) 编写一个用于计算某账户资金在 5%~10% 的利率下 (含 5% 和 10%), 经过 10 年以后所得存款总额的应用程序 (参见图 10.29)。在应用程序中, 用户必须提供初始的本金。

- a) **将模板复制到工作目录中** 将 C:\Examples\Tutorial10\Exercises\ComparingRates 目录复制到 C:\SimplyJava 目录中。
- b) **打开模板文件** 在自己的文本编辑器中打开 ComparingRates.java 文件。
- c) **自定义 JTextArea** 必须自定义可显示 10 年后利率和资金总数额的 JTextArea。此 JTextArea 的名字为 resultJTextArea。在第 58 行, 插入将该组件的 bounds 属性设置为 20, 85, 260, 120 的代码。在第 59 行, 插入将 resultJTextArea 的 editable 属性设置为 false 的代码。

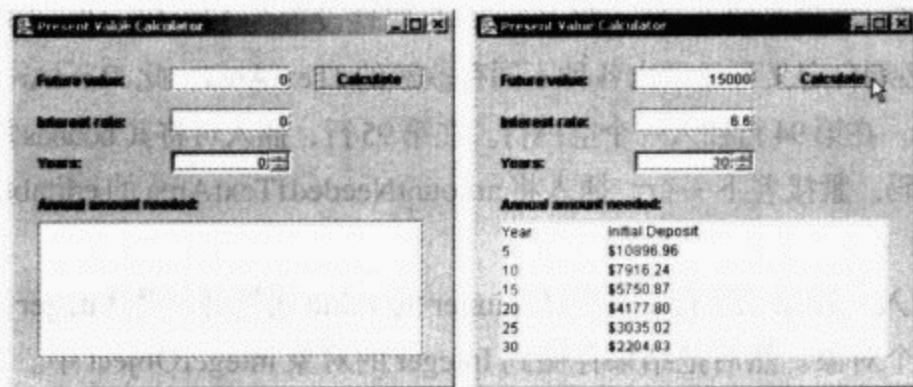


图 10.28 当前值计算器 GUI

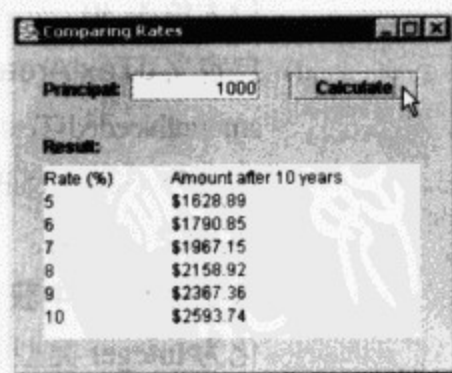


图 10.29 对比率 GUI

- d) **完成 for 语句的首部** 在第 98 行, 会看到一条只含有两个分号的 for 语句首部。在第一个分号之前, 声明变量 rate (将作为我们的计数器) 并初始化为 5。在第二个分号之前, 输入一个可导致 for 语句不断循环直至该计数器到达 10 的循环-继续条件。在第二个分号之后, 输入计数器的自增运算, 使 for 语句可从百分率 5 增长到百分率 10。

- e) **计算 10 年后的资金总额** 下面, 将针对不同的利率, 计算出 10 年后的存款总额。为完成这项工作, 需要在 for 语句中实现下面的公式:

$$a = p(1 + r)^n$$

其中:

p 为初始的投资额 (本金)

r 为年利率

n 为年限数

a 为第 n 年年底的投资额

在第 100 行至第 101 行, 利用 Math.pow 方法 (以及第 94 行至第 95 行已定义好的变量) 在 for 语句中计算出当前利率下 10 年后的价值。使用必要地造型运算符 (double), 以保证 Math.pow 的第一个参数为一个 double 值。使用两行语句是为了增强代码的清晰性。在第 102 行至第 103 行, 利用 append 方法在应用程序 JTextArea 的内部输出计算出的结果值。需使用第 137 行已创建好的 DecimalFormat 对象 (dollars)。同样也使用了两行语句, 是为了增强代码的清晰性。

- f) **保存应用程序** 保存修改后的源代码文件。
- g) **打开命令提示符窗口改变目录** 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\ComparingRates 进入到当前的工作目录中。
- h) **编译应用程序** 通过键入 javac ComparingRates.java, 编译该应用程序。
- i) **运行完成后的应用程序** 若应用程序能正确编译, 通过键入 java ComparingRates 来运行它。输入本金, 然后点击 Calculate JButton。查看结果以确保显示出正确的利率, 并保证预期的价值也是正确的。
- j) **关闭应用程序** 点击关闭按钮关闭正在运行的应用程序。
- k) **关闭命令提示符窗口** 点击命令提示符窗口的关闭按钮将其关闭。

10.13 (可验证输入的利息计算器应用程序) 改进本教程中所创建的利息计算器应用程序, 使其具有错误校验的功能。该应用程序能够测试出用户是否输入了无效的本金和利率值。假若输入了一个无效值, 应在一个消息对话框中显示出一条消息。图 10.30 演示了这个可处理无效数据输入的应用程序的执行过程。

- a) **将模板复制到工作目录中** 将 C:\Examples\Tutorial10\Exercises\InterestCalculatorEnhanced 目录复制到 C:\SimplyJava 目录中。
- b) **打开模板文件** 在自己的文本编辑器中打开 InterestCalculator.java 文件。

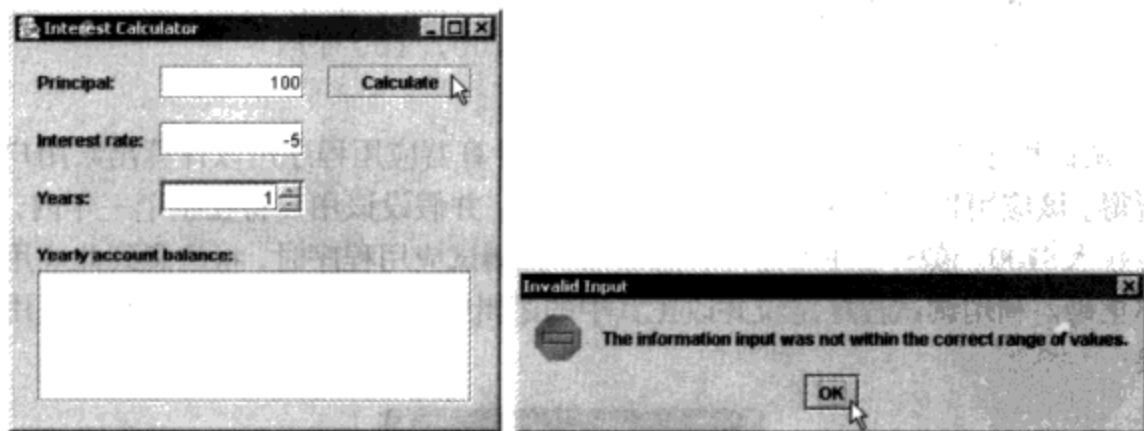


图 10.30 具有错误校验功能的利息计算器应用程序

- c) **自定义可处理无效输入的 calculateJButtonActionPerformed 方法** 在第 143 行, 为 if 语句加入一个条件, 该条件在本金或利率为负数或者是当利率超过 10 时, 便会返回一个 true。
- d) **显示用以提示错误的消息** 在第 145 行至第 147 行, 显示一个如图 10.30 所示的消息对话框。使用三行语句是为了增强代码的清晰性。
- e) **保存应用程序** 保存修改后的源代码文件。

- f) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\InterestCalculatorEnhanced` 进入到当前的工作目录中。
- g) 编译应用程序 通过键入 `javac InterestCalculator.java`, 编译该应用程序。
- h) 运行完成后的应用程序 若应用程序能正确编译, 通过键入 `java InterestCalculator` 来运行它。输入本金、利率和年限数, 然后点击 Calculate JButton。确保所有的输入都是有效的。通过查看结果以确保显示出正确的年限数, 并保证所得到的存款额也是正确的。接着, 修改已输入的值, 使之成为无效输入, 然后点击 Calculate JButton, 确保可出现一个包含了正确文本内容的消息对话框。
- i) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- j) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

10.14 (说出这段代码的作用) 下列代码的执行结果是什么? 假设 `power` 和 `number` 均已声明为 `int`。

```
1 power = 5 ;
2 number = 10 ;
3 mysteryJTextArea.setText( "" );
4
5 for ( int counter = 1 ; counter <= power; counter++ )
6 {
7     mysteryJTextArea.append( Math.pow( number, counter ) + "\n" );
8 }
```

10.15 (找出代码中的错误) 寻找下列代码中的错误:

- a) 此段代码将按照降序的顺序在一个 JTextArea 中显示出从 100 到 1 之间的所有整数。

```
1 String output;
2
3 for ( int counter = 100; counter >= 1 ; counter++ )
4 {
5     output += counter + "\n";
6 }
7
8 displayTextArea.setText( output );
```

- b) 下列代码将把一个 JSpinner 的范围设置为 2~100 之间的偶数。此 JSpinner 的初始值为 2。

```
1 yearsJSpinner = new JSpinner(
2     new SpinnerNumberModel( 100, 2 , 100, 1 ) );
3 yearsJSpinner.setBounds( 2 , 2 , 100, 100 );
4 contentPane.add( yearsJSpinner );
```

10.16 (使用调试程序)(存款计算器应用程序)存款计算器应用程序用以计算出某用户存款一年后的全部所得。该应用程序从用户取得最初的存款额, 并假设该用户将在整个一年内, 每月定期地向该账号存入 \$100。该账号并无任何利息加入。在测试应用程序时, 将注意到此应用程序的计算结果并不正确。利用调试程序定位并改正其中的逻辑错误。图 10.31 显示的是该应用程序正确的输出结果。

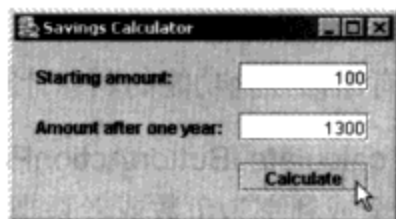


图 10.31 存款计算器应用程序的正确输出

- a) 将模板复制到工作目录中 将 `C:\Examples\Tutorial10Exercises\Debugging\SavingsCalculator` 目录复制到 `C:\SimplyJava` 目录中。

- b) 打开模板文件 在自己的文本编辑器中打开 SavingsCalculator.java 文件。
- c) 运行应用程序 输入 java SavingsCalculator 运行存款计算器应用程序。输入 100 作为初始的资金额, 然后点击 Calculate JButton。注意到一年后该存款额变成了 1200, 然而, 正确的结果应为 1300 (参见图 10.31)。
- d) 利用 -g 选项进行编译 关闭应用程序 (但保留命令提示符窗口), 通过键入 javac -g SavingsCalculator.java 编译该应用程序。
- e) 启动调试程序 通过键入 jdb 启动调试程序。
- f) 寻找并更正错误 利用前面教程中所学到的调试技巧, 确定该应用程序中存在的逻辑错误。修改应用程序使之显示出正确的结果。
- g) 保存应用程序 保存修改后的源代码文件。
- h) 编译应用程序 通过键入 javac SavingsCalculator.java, 编译该应用程序。
- i) 运行完成后的应用程序 若此应用程序能正确编译, 键入 java SavingsCalculator 来运行它。通过使用不同的输入对该应用程序进行测试, 并确保能够得到正确的存款额。
- j) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- k) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

挑战题

10.17 (加薪计算器应用程序) 开发一个应用程序, 计算出某员工在指定的年限内, 每一年所得的薪水总额。假设该员工的周薪为 \$500, 且每年只能加薪一次。用户需在应用程序中指定工资的增长幅度 (将按照每年总额的百分比进行计算) 及所赚取薪水的年限。该应用程序的运行结果如图 10.32 所示。

- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial10\Exercises\PayRaise 目录复制到 C:\SimplyJava 目录中。
- b) 打开模板文件 在自己的文本编辑器中打开 PayRaise.java 文件。
- c) 自定义 Amount of raise(in%):JSpinner 必须自定义用于显示加薪百分比的 JSpinner。此 JSpinner 的名称为 raiseJSpinner。用户所指定的百分比范围只能在范围 3%~8% 之间。修改第 53 行, 使 raiseJSpinner 的初始值为 3, 最小值为 3, 最大值为 8, 步长为 1。在第 54 行, 插入可将其 bounds 属性设置为 170, 25, 70, 22 的代码。

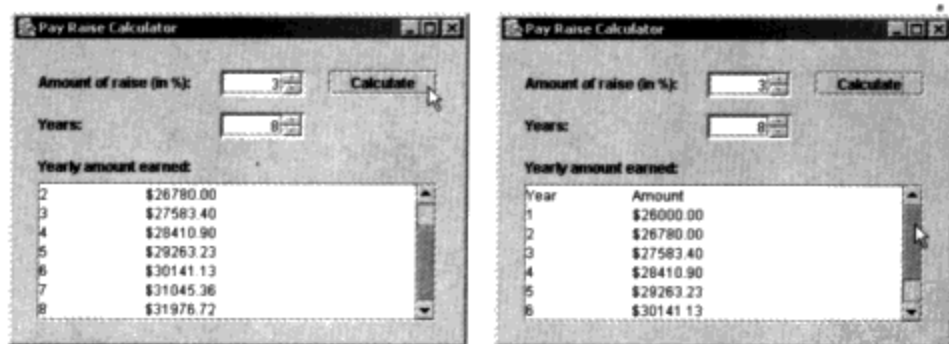


图 10.32 加薪应用程序

- d) 自定义 Years:JSpinner 必须自定义可用于显示其范围在 1~50 之间年限数的 JSpinner。此 JSpinner 的名称为 yearsJSpinner。修改第 65 行使 yearsJSpinner 的初始值为 1, 其最小值为 1, 其最大值为 50, 步长为 1。在第 77 行, 插入可将其 bounds 属性设置为 170, 60, 70, 22 的代码。
- e) 自定义 JTextArea 必须自定义出 amountEarnedJTextArea, 它将按照多行形式显示出每年所赚取的薪水。在第 77 行, 插入可将 amountEarnedJTextArea 的 editable 属性设置为 false 的代码。
- f) 自定义 JScrollPane 必须自定义包含 amountEarnedJTextArea 的 JScrollPane, 使这一 amountEarnedJTextArea 在必要时显示出滚动条。修改第 80 行, 将 amountEarnedJTextArea 添加到应用程序里的 JScrollPane 中。使用了两行语句是为了增强代码的清晰性。在第 82 行, 插入可将其 bounds 属性设置为 20, 120, 330, 115 的代码。

- g) 从 `raise JSpinner` 中取得输入 在第 117 行至第 118 行, 访问 `raiseJSpinner` 的 `value` 属性并利用 `Integer`, 将结果转化成 `Integer` 类型的对象。之后, 将结果存储到 `Integer` 的对象 `integerRaiseObject` 中。使用两行语句是为了增强代码的清晰性。在第 119 行, 取出 `integerRaiseObject` 中的 `int` 数据, 并将该数据存储到 `int` 型的变量 `rate` 中。
- h) 从 `years JSpinner` 中取得输入 在第 121 行至第 122 行, 访问 `yearsJSpinner` 的 `value` 属性并利用 `Integer`, 将结果转化成 `Integer` 类型的对象。之后, 将结果存储到 `Integer` 的对象 `integerYearsObject` 中。在第 123 行, 取出 `integerYearsObject` 中的 `int` 数据, 并将该数据存储到 `int` 型的变量 `years` 中。在第 123 行的后面插入一个空白行, 使上述代码行与其他的代码分开。
- i) 完成一个 `for` 语句首部 在第 129 行, 会看到一条只含有两个分号的 `for` 语句首部。在第一个分号之前, 声明变量 `counter` 并初始化为 1。在第二个分号之前, 输入一个可导致 `for` 语句不断循环直至 `counter` 到达所输入年限数的循环-继续条件。在第二个分号之后, 输入 `counter` 的自增运算, 使该 `for` 语句能够针对每一个年限数执行一次相应的计算。
- j) 计算加薪 在第 131 行, 将工资 (已在第 115 行上被设置为 500) 乘以一年的总周数, 然后, 把结果存储在一个名为 `amount` 的 `double` 型变量中。通过第 132 行至第 133 行, 把结果可显示在 `JTextArea` 中。使用了两行语句是为了增强代码的清晰性。在第 135 行, 是为紧接着的年限计算出相应的工资额, 并把结果存储到变量 `wage` 中。要完成这项工作, 可将 1 加到所递增的百分比值中, 然后, 再将结果乘以 `wage` 中的当前值。
- k) 保存应用程序 保存修改后的源代码文件。
- l) 打开命令提示符窗口改变目录 选择 `Start → Programs → Accessories → Command Prompt` 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\PayRaise` 进入到当前的工作目录中。
- m) 编译应用程序 通过键入 `javac PayRaise.java`, 编译该应用程序。
- n) 运行完成后的应用程序 若应用程序能正确编译, 通过键入 `java PayRaise` 来运行它。输入一个加薪百分比和工资增长的年限数, 然后点击 `Calculate JButton`。注意观察其中的结果, 确保能够显示出正确的年限数, 并保证未来的工资值也是正确的。同时, 考察 `JSpinner`, 确保它们为用户提供出了正确的数据。
- o) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- p) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。



教程 11 门禁系统应用程序

介绍 switch 多向选择语句，Date 及 DateFormat 类

教学目标

在本教程中，读者将学到以下内容：

- switch 多向选择语句
- case 标签
- 显示日期与时间
- JPasswordField
- 确定系统当前日期与时间的 Date
- 格式化日期与时间的 DateFormat

在教程 6 中，曾学习过可用做选择或忽略某个单向语句（或语句序列）的 if 控制语句，该语句也因此被称为单向选择语句。与此同时，还学习了能应用于多条语句（或语句序列）并根据某个条件来进行选择的 if...else 控制语句，该语句则被称为双向选择语句。在本教程中，将学习 switch 多向选择语句，通过它在多个可能的操作（或操作序列）之间完成选择操作。

11.1 探试门禁系统应用程序

在本教程中，将利用 switch 多向选择语句创建一个门禁系统应用程序。该应用程序必须满足下面的需求：

应用程序需求分析

某制药公司希望在配有设备的实验室外安装一套门禁系统。只有得到授权并取得保护密码的人员才允许进入该实验室。以下便是一些有效的保护密码（也称为访问码）及所代表的雇员小组：

保护密码	雇员小组
1645	技术员
8345	管理员
9998, 1006~1008	科学家

当输入保护密码时，应使该保护密码对任何可能站在门禁系统旁边的人员都是不可见的。而所输入的每一个保护密码，其请求要么是被允许要么是被拒绝。所有试图进入实验室的请求都将显示在位于小键盘下侧的一个屏幕当中。即如果请求被允许，则相应的日期、时间及其所属的小组名称（科学家、管理员，等等）均会显示在屏幕当中；如果请求被拒绝，则相应的日期、时间以及一条信息“Access Denied”也会显示在屏幕当中。此外，任何雇员还可通过输入访问码 7，8 或 9 来寻求安检员的帮助，并且在上述访问码输入完后屏幕中会显示相应的日期、时间以及一条消息“Restricted Access”，表示已收到该请求。

我们将以这个完成后的应用程序的探试作为开始。之后，学习一些 Java 技术并最终创建一个属于自己的应用程序。



探试门禁系统应用程序

1. 定位到完成后的应用程序 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\Examples\Tutorial11\CompletedApplication\SecurityPanel` 将目录改变到这个完成后的门禁系统应用程序的目录下面。
2. 运行门禁系统应用程序 键入 `java SecurityPanel` 运行该应用程序（参见图 11.1）。可以看到，这个 GUI 小键盘看起来非常像一个实际的小键盘（若有可能，尽量去模拟现实世界中出现的情形）。



GUI 设计提示

如果进行开发的 GUI 被用于模拟现实世界中的某个物体的话，则相应的 GUI 设计就应该模拟出该物体的物理外观。

3. 键入一个无效保护密码 利用小键盘输入一个无效密码 1212（参见图 11.2）。在 Security code:JLabel 的右侧为一个 JPasswordField 组件，它会把 GUI 小键盘上所输入的保护密码中的每一个数字以一个星号来显示。JPasswordField（类似于 JTextField）将不会显示出应用程序用户所输入的实际字符。JPasswordField 利用星号作为实际输入字符的替代物，因而确保了密码的安全性。通过一些特定的字符来隐藏保护密码，因此，其他人员将无法看到相应的访问码。这就像我们在登录自己的计算机时，会常常碰到类似的情形。

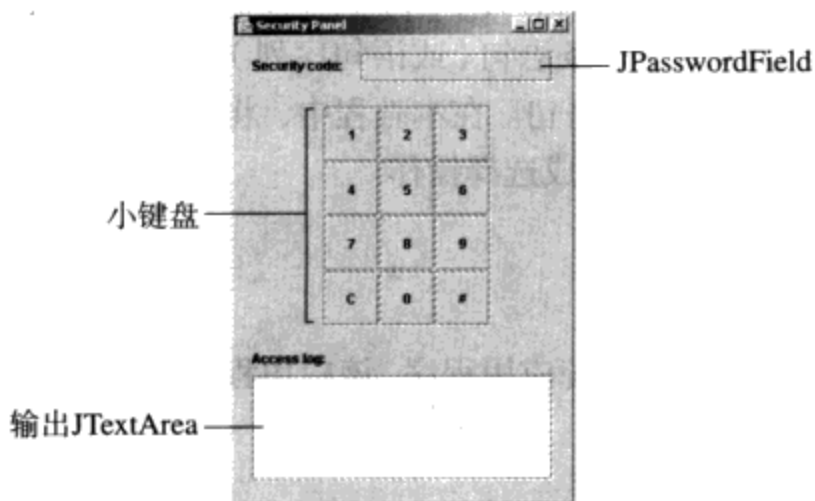


图 11.1 门禁系统应用程序

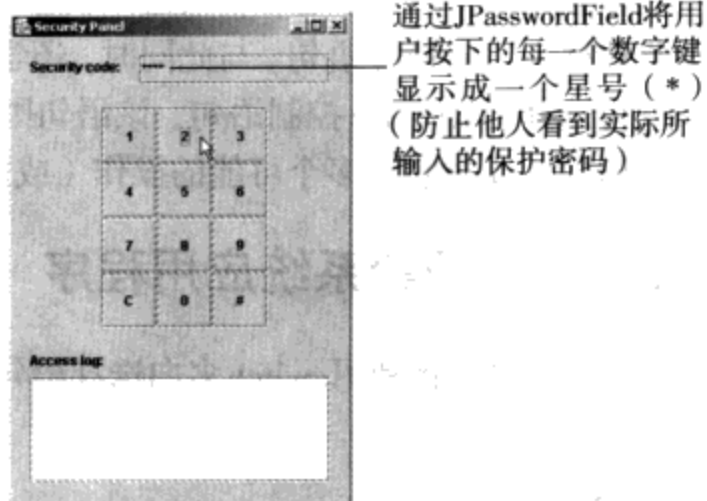


图 11.2 在 Security code:JPasswordField 中显示星号

4. 输入完安检码以后，点击用以处理应用程序输入的 #JButton 相应结果应显示在位于该应用程序低端的 JTextArea 中。可以看到，一条用以表明请求被拒绝的信息出现在了 JTextArea 中（如图 11.3 所示）。注意，当 #JButton 被按下时，JPasswordField 中的内容同时也会被清除掉。
5. 使用 C JButton 按下一些数字键，然后，点击 C JButton 便可清除 Security code:JPasswordField 中的内容。这时，所有曾在 JPasswordField 中显示的星号都将消失。通常，由于用户会在按键或者点击 JButton 时出现差错，因而，通过使用 C JButton 可让用户在出现错误的时候能够及时地清理掉 JPasswordField 中的内容并重新开始输入。



GUI 设计提示

利用 JPasswordField 屏蔽密码或其他敏感信息。

6. 输入有效保护密码 利用小键盘输入 1006，然后点击 #JButton。可以看到，JTextArea 中出现了第二条信息，参见图 11.4。
7. 关闭正在运行的应用程序 点击关闭按钮关闭正在运行的应用程序。
8. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

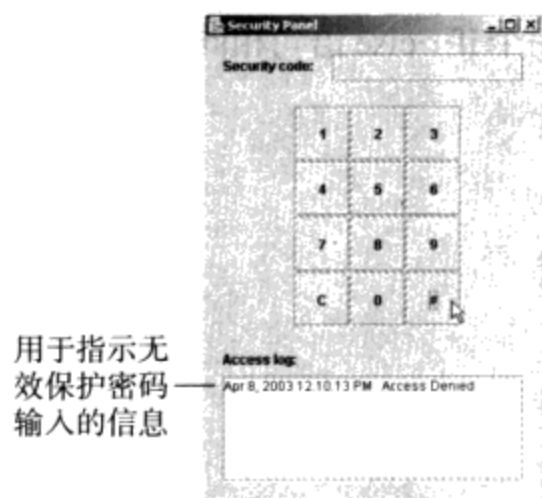


图 11.3 显示请求拒绝信息的门禁系统

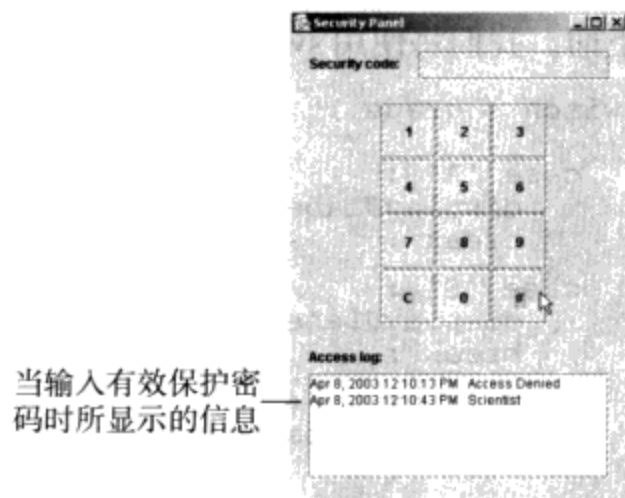


图 11.4 确认有效保护密码输入的门禁系统应用程序

11.2 介绍 switch 多向选择语句

在本节中，我们将学习如何使用 switch 多向选择语句。为了能够更好地进行比较，首先考虑一个嵌套的 if...else 语句，它将根据学生的成绩执行多个选择，并利用 JLabel 将所得到的选择利用一条文本消息进行显示：

```
if ( grade == 'A' )
{
    displayJLabel.setText( "Excellent!" );
}
else if ( grade == 'B' )
{
    displayJLabel.setText( "Very good!" );
}
else if ( grade == 'C' )
{
    displayJLabel.setText( "Good." );
}
else if ( grade == 'D' )
{
    displayJLabel.setText( "Poor." );
}
else if ( grade == 'F' )
{
    displayJLabel.setText( "Failure." );
}
else
{
    displayJLabel.setText( "Invalid grade." );
}
```

该条语句可在多个 grade 值之间进行选择并产生一个适当的输出。然而，如果使用 switch 语句，可将每一种情况，如：

```
if ( grade == 'A' )
```

简化为：

```
case 'A':
```

在本例中，grade 是一个 char 类型的变量，这种类型属于 Java 中 8 个基本类型中的一个（参考附录 F 中有关 Java 基本类型的一个完整列表）。char 类型变量的值是一个字符常量（或叫字符面值），可通过位于一对单引号之间的单个字符或转义序列进行表示。字符常量包括字母（如 'A'）、数字（如 '5'）、特定字符（如 ',' ——逗号）、空格（如 ' ' ——空格）以及转义序列（如 '\n' ——换行字符），等等。

下面,我们将使用 switch 多向选择语句,完成与前一个 if...else 语句相同的功能:

```
switch ( grade )
{
    case 'A' :
        displayJLabel.setText( "Excellent!" );
        break ;
    case 'B' :
        displayJLabel.setText( "Very good!" );
        break ;
    case 'C' :
        displayJLabel.setText( "Good." );
        break ;
    case 'D' :
        displayJLabel.setText( "Poor." );
        break ;
    case 'F' :
        displayJLabel.setText( "Failure." );
        break ;
    default :
        displayJLabel.setText( "Invalid grade." );
}
```

switch 语句起始于关键字 switch, 后跟一个位于圆括号内的控制表达式和一个左花括号, 并最终以一个右花括号作为结束。上述 switch 语句中包含了 5 个 case 标签及一个可选用的 default case, default case 将在控制表达式的值与其他任何 case 之间无法匹配时执行。每一个 case 标签将包含一个关键字 case, 一个常量表达式和一个冒号。常量表达式可以是文本字面值, 如 'A' (属于 char 类型) 或者是一个整数值, 如 707, 但不能是一个浮点值 (如 9.9)。常量表达式还可以是包含了某个字符或整数常量的一个变量 (即一个 final 变量)。只有属于 char, byte, short 和 int 类型的值才允许在 switch 语句中进行测试。尽管一条 switch 语句可拥有任意数量的 case 标签, 但它最多只能有一个 default case, 而且两个 case 不能指定为同一个表达式。



好的编程习惯

在 switch 语句中每一个 case 的前后放置一个空白行将改善代码的可读性。



好的编程习惯

在 case 语句体中进行缩进, 将有助于改善代码的可读性。



常见编程错误

若两个或两个以上的 case 中拥有完全相同的常量表达式, 则将出现一个语法错误。



常见编程错误

在 case 中忘记使用 break 语句常常会导致一个逻辑错误的出现。

图 11.5 中给出了上述 switch 多向选择语句的 UML 活动图。需要计算的第一个警戒条件为 `grade == 'A'`。如果该条件为 true, 则显示文本 "Excellent!", 而且该 case 末尾的 break 语句会把程序的控制转移至整个 switch 语句后的第一条语句上。如果计算该条件的结果为 false (即警戒条件 `grade != 'A'` 为 true), 则继续测试下一个条件, `grade == 'B'`。如果计算该条件的结果为 true, 则显示文本 "Very good!", 且该 case 末尾的 break 语句会把程序的控制转移至整个 switch 语句后的第一条语句上。如果计算该条件的结果为 false (即警戒条件 `grade != 'B'` 为 true), 则又继续测试下一个条件。这一过程将一直进行下去直至找到一个相匹配的 case, 或者是进行到最后一个警戒条件 `grade != 'F'` 且计算的结果为 false 时停止。如果是后者, 则执行 default case 的语句体, 即显示文本 "Invalid grade"。此时, 应用程序将继续执行位于 switch 语句后的第一条语句。如果控制表达式的值没有找到相匹配的 case 且不存在 default case, 则 switch 中将不会执行任何的语句。



错误预防提示

总是在 switch 语句中包含一个 default case 将确保所有不合适的值得到处理。

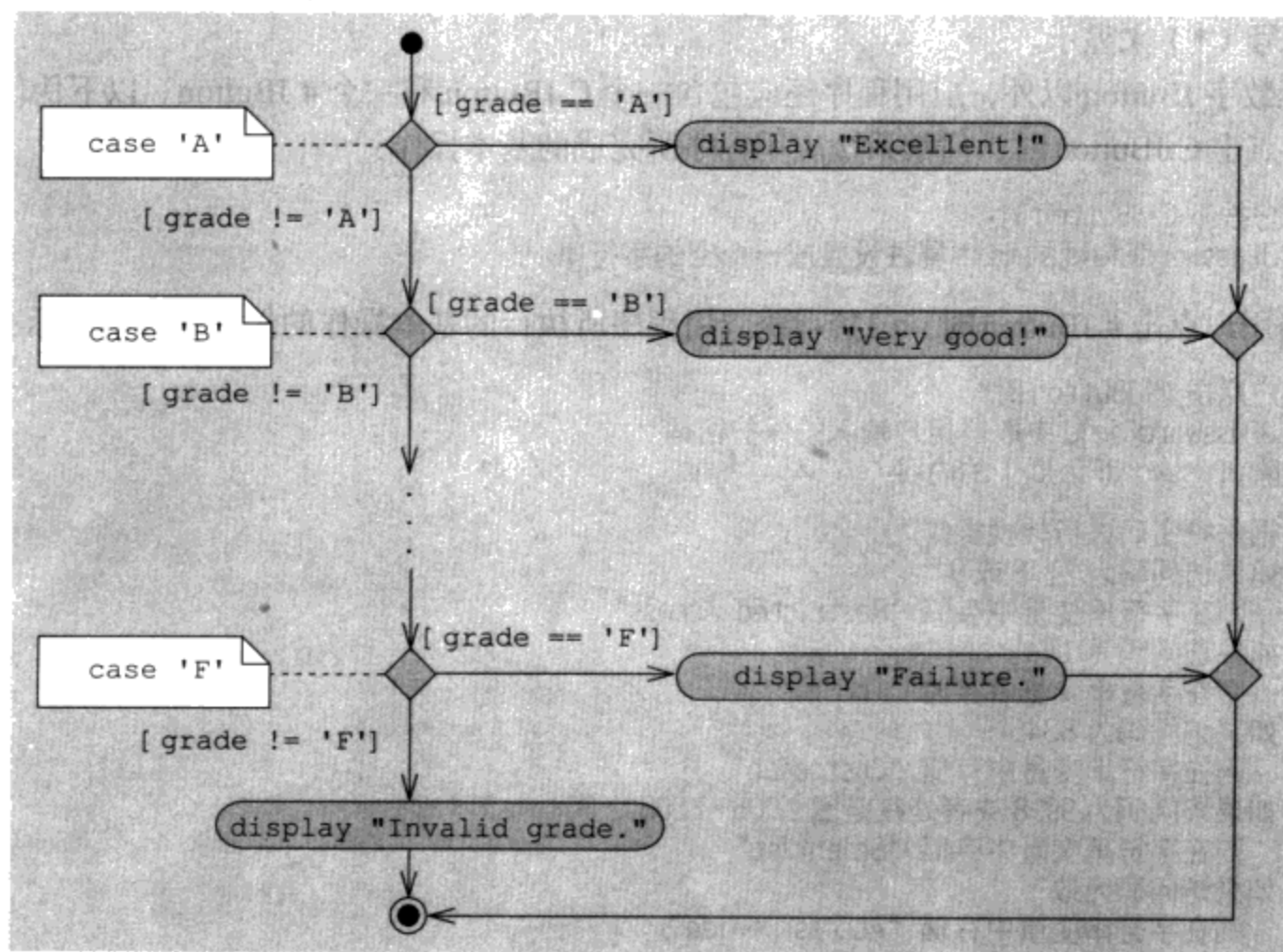


图 11.5 switch 多向选择语句的 UML 活动图

如果控制表达式的值能与一个 case 语句相匹配，而且该 case 语句并未以一条 break 语句结束，那么，程序控制将继续执行（下落至）下一个 case 中的语句。这一过程将持续下去直至遇到一个 break 语句，或者是到达 switch 语句的末尾。

自测题

- switch 是一条 _____ 选择语句。
 - 单向
 - 双向
 - 多向
 - 以上答案均不对
- 何时执行 default case?
 - 每执行一条 switch 语句时
 - 当存在一些能与控制表达式相匹配的 case 时
 - 当不存在能与控制表达式相匹配的 case 时
 - 以上答案均不对

答案：1) c 2) c

11.3 创建门禁系统应用程序

门禁系统应用程序中将包含 10 个用以显示阿拉伯数字的 JButton（数字 JButton）。我们需为每一个 JButton 创建出一个可以执行的方法。这些 JButton 便构成了一个 GUI 小键盘。以下伪代码描述的便是有关这些数字 JButton 的相关操作：

当用户点击一个数字 JButton 时

将正确的阿拉伯数字追加到 JPasswordField 的文本中

用户将利用这些数字 JButton, 输入阿拉伯数字并把相应的数字追加到 JPasswordField 中的文本的后面。注意, 每个数字都会在相应 JButton 被点击时显示出来; 然而, 所输入的数字在 JPasswordField 中将以星号 (*) 来显示。

除了数字 JButton 以外, 应用程序还应包含一个 C JButton 和一个 # JButton。以下伪代码描述了当用户点击 C JButton 时, 门禁系统应用程序所完成的基本操作:

当用户按下 C JButton 时
 将 JPasswordField 的 text 属性设置成一个空的字符串

而当用户点击 # JButton 时, 门禁系统应用程序所执行的基本操作的伪代码如下所示:

当用户点击 # JButton 时
 从 JPasswordField 中取得用户输入的保护密码
 清除 JPasswordField 中的内容

 根据保护密码进行转换操作
 如果访问码为 7, 8 或 9
 则在字符串变量中存储 "Restricted Access"
 如果访问码为 1645
 则在字符串变量中存储 "Technician"
 如果访问码为 8345
 则在字符串变量中存储 "Custodian"
 如果访问码为 9998 或者处在范围 1006~1008 之间
 则在字符串变量中存储 "Scientist"
 如果访问码无效
 则在字符串变量中存储 "Access Denied"

将包含当前时间以及字符串变量内容的一条消息显示在 JTextArea 中

既然我们已对门禁系统应用程序进行了探试并研究了它的伪代码表示; 下面将使用一张 ACE 表, 帮助我们最终把这个伪代码转换成 Java 的实现。图 11.6 中列出了该应用程序中相应的操作、组件以及事件, 以最终完成属于自己的这一应用程序。第一行表示将使用 JLabel 对 JPasswordField 和 JTextArea 组件进行标记。第二行引入了数字小键盘上的 JButton。当其中一个 JButton 被点击时, 该 JButton 上的值将被追加到 JPasswordField 的 text 属性的文本中。下一行表示将用 securityCodeJPasswordField 存储用户所输入的保护密码。紧接着的两行表示, 用户可通过点击 JButton (clearJButton)清除 securityCodeJPasswordField 中的内容(通过使用一个清除 JPasswordField 内容的方法实现)。再下一行, 指明了用户可通过点击 JButton(enterJButton)来提交保护密码, 从而使应用程序能够确定出该保护密码是否有效。剩余行表示的是当点击 enterJButton 时, 需执行的任务。



操作	组件	事件
标记应用程序中的组件	securityCodeJLabel, accessLogJLabel	当应用程序运行时
	zeroJButton, oneJButton, twoJButton, threeJButton, fourJButton, fiveJButton, sixJButton, sevenJButton, eightJButton, nineJButton	当用户点击一个数字 JButton 时
将正确的阿拉伯数字追加到 JPasswordField 的文本中	securityCodeJPasswordField	
	clearJButton	当用户点击 C JButton 时
将 JPasswordField 的 text 属性 设置成一个空的字符串	securityCodeJPasswordField	
	enterJButton	当用户点击 # JButton 时

从 JPasswordField 中取得用户输入的保护密码 securityCodeJPasswordField

(续表)

操作	组件	事件
清除 JPasswordField 中的内容	securityCodeJPasswordField	
根据保护密码进行转换操作		
如果访问码为 7, 8 或 9	message(String)	
则在字符串变量中存储 "Restricted Access"		
如果访问码为 1645	message(String)	
则在字符串变量中存储 "Technician"		
如果访问码为 8345	message(String)	
则在字符串变量中存储 "Custodian"		
如果访问码为 9998 或处在范围 1006~1008 之间	message(String)	
则在字符串变量中存储 "Scientist"		
如果访问码无效	message(String)	
则在字符串变量中存储 "Access Denied"		
将包含当前时间以及字符串变量内容的一条消息显示在 JTextArea 中	accessLogJTextArea	message(String)

图 11.6 门禁系统应用程序的 ACE 表

下面，将自定义一个用于存储用户访问码的 JPasswordField 组件。在本教程的后面，还将通过一个 switch 语句构建出这个门禁系统应用程序。

自定义 JPasswordField 组件

- 1. 将模板复制到工作目录中 将 C:\Examples\Tutorial11\TemplateApplication\SecurityPanel 目录复制到 C:\SimplyJava 目录中。
- 2. 打开门禁系统应用程序的模板文件 在自己的文本编辑器中打开模板文件 SecurityPanel.java。
- 3. 自定义 JPasswordField 插入图 11.7 中第 57 行至第 58 行的代码。第 57 行是将 securityCodeJPasswordField 的 bounds 属性设置为 114, 16, 172, 26。JPasswordField 中显示的文本会通过星号字符 (*) 进行隐藏，或称为屏蔽。可以看到，键入的实际文本的确没有直接显示出来，而是通过回应字符 (也称屏蔽字符) 进行显示。JPasswordField 的 password 属性将包含用户所输入的文本。例如，若用户输入 5469，则 JPasswordField 中将显示 ****，而实际的 "5469" 将存储在它的 password 属性中。可利用 JPasswordField 的 setEchoChar 方法并通过为其传递一个字符常量来改变所显示的回应字符。在这个应用程序中，当用户按下数字小键盘上的按钮时，便可输入相应的保护密码，而 JPasswordField 中仅仅只是显示得到屏蔽的保护密码。第 58 行上的语句，可用于防止用户修改 JPasswordField 中的文本，这样做的目的是，迫使用户只能同数字小键盘进行交互，而这一功能是通过将 JPasswordField 的 editable 属性设置为 false 来实现的。
- 4. 保存应用程序 保存修改后的源代码文件。
- 5. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\SecurityPanel 进入到当前的工作目录中。
- 6. 编译应用程序 通过键入 javac SecurityPanel.java 编译该应用程序。
- 7. 运行应用程序 若此应用程序能正确编译，通过键入 java SecurityPanel 来运行它。图 11.8 中显示了更新后的应用程序的运行结果。试着去按一些 JButton。可以看到，什么也没有发生，这是因为到目前为止读者仍未编写点击 JButton 时所执行的方法。
- 8. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- 9. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

设置JPasswordField
的bounds属性和
editable属性

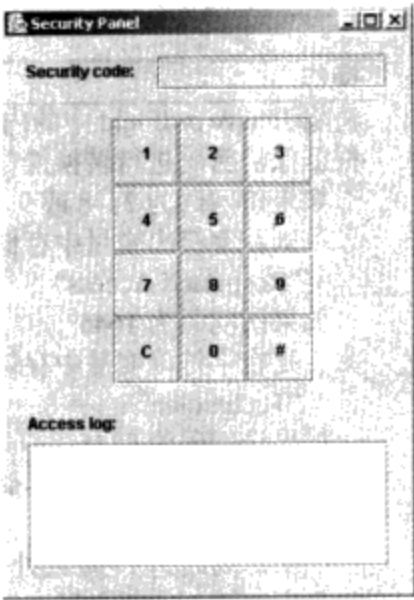
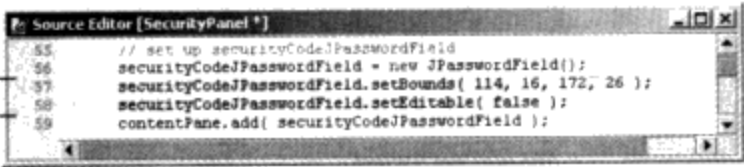


图 11.7 设置 securityCodeJPasswordField 的 bounds 属性和 editable 属性 图 11.8 自定义 JPasswordField 以后的门禁系统应用程序

现在，我们已经设计出了该应用程序的GUI，下面，将对方法中所使用的一些变量执行初始化操作并实际地编写出一条 switch 语句。该语句将按照所输入的代码，确定用户的准入级别。

向应用程序中添加一条 switch 语句

- 1. 声明并初始化局部变量 将图 11.9 中第 379 行至第 385 行添加到 enterJButtonActionPerformed 方法中。第 379 行声明了一个String 变量message，利用这一变量并根据所输入的访问码，存储一条显示给用户的信息。第 382 行至第 383 行通过调用 securityCodeJPasswordField 的 getPassword 方法来获取用户输入的密码。将该密码传递给方法String.valueOf，从而将其转换成一个String。接着，再利用Integer.parseInt方法将其转化为一个int。最终，便把这个密码的int 值赋给变量accessCode。第 385 行用做清除Security code:JPasswordField 中的内容，以使用户能够重新输入新的代码。

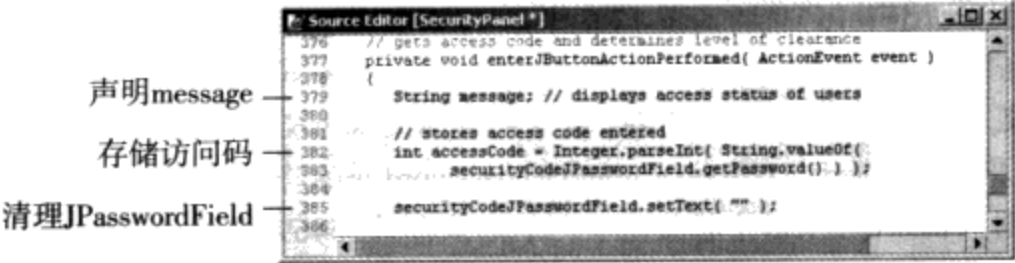


图 11.9 存储访问码以及清理 Security code:JPasswordField 中的内容

- 2. 在 enterJButtonActionPerformed 中加入一条 switch 语句 将图 11.10 中第 387 行至第 390 行添加到 enterJButtonActionPerformed 方法中。从第 387 行开始，为一条 switch 语句，其中将包含一个控制表达式 accessCode（作为用户输入的访问码）。切记，该表达式（accessCode 的值）将按照从上到下的顺序与每一个 case 进行比较。如果能找到一个相匹配的 case，则执行该 case 的语句体，并通过这一 case 语句体中的 break 语句将程序的控制转移至位于右花括号后的第一条语句上。如果该 switch 语句中含有一个 default case 且控制表达式将不能匹配任何的 case 语句，那么，该 default case 语句体中的语句将得到执行。假若无法找到匹配，且 switch 中也不包含 default case，则会跳过整个 switch 语句体，即应用程序将继续执行位于 switch 后的下一条语句。第 390 行为一个终止 switch 语句的右花括号。

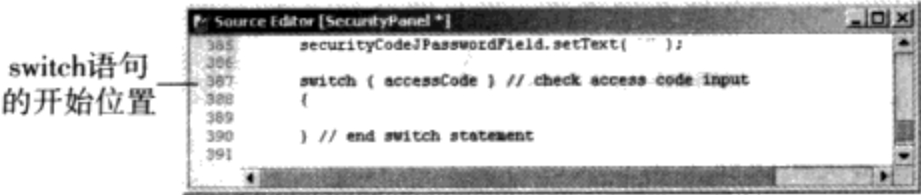


图 11.10 在方法中添加一条 switch 语句

3. 在 switch 语句中添加 case 将图 11.11 中第 389 行至第 394 行插入到 switch 语句中。在第 390 行至第 392 行中指定了三个 case 标签。这意味着如果这三个 case 标签中的任何一个可与控制表达式的值相匹配的话，则第 393 行至第 394 行都将得到执行。例如，如果 accessCode 的值为 7, 8 或 9，则第 393 行至第 394 行上的代码会得到执行，即字符串 "Restricted Access" 将被赋值给 message。



图 11.11 在 switch 语句中添加 case 标签

4. 为剩余的访问码指定 case 将图 11.12 中第 396 行至第 412 行添加到 switch 语句的语句体内。第 397 行上的 case 标签将用于确定 accessCode 的值是否等于 1645。若用户的确输入的是这个访问码，则该 case 的语句体会把 message 的值设置为 "Technician"。下一个 case 标签（参见第 402 行）用于检查 accessCode 的值是否为 8345。如果用户的确输入的是这一访问码，则此 case 的语句体将把 message 的值设置为 "Custodian"。再下一个 case 标签（第 407 行至第 410 行）将用于确定 accessCode 的值是否为 9998, 1006, 1007 或 1008。如果输入的访问码为上述 4 个中任何一个的话，则此 case 语句体将把 message 的值设置为 "Scientist"。

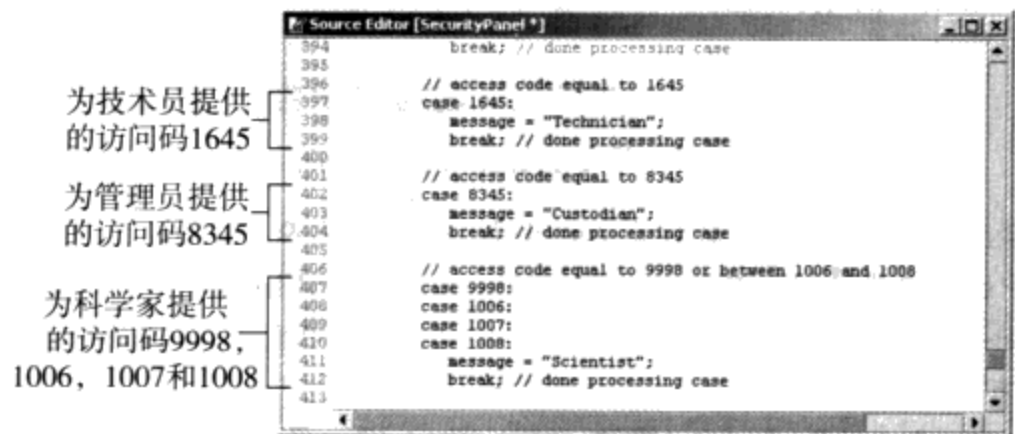


图 11.12 完成 switch 语句

5. 为 switch 语句添加一个 default case 将图 11.13 中第 414 行至第 416 行添加到 switch 语句中。在这些行中，包含了一个可选的 default case，它是在控制表达式不能匹配任何的 case 时开始执行的。当 default case 出现在语句体的末尾时，并不需要为其添加 break 语句。本例中，这个 default case 的语句体是把 message 的值设置为 "Access Denied"。

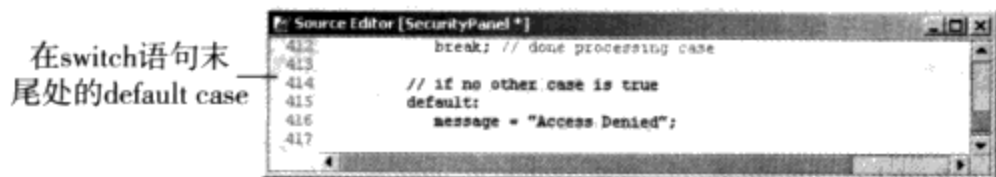


图 11.13 为 switch 语句添加一个 default case



好的编程习惯

习惯上是将 default case 放置在 switch 语句体的末尾处。

6. 在 JTextArea 中显示结果 将图 11.14 中第 420 行至第 423 行添加到 switch 语句中。第 421 行将用于取得一个 DateFormat 对象（利用 DateFormat 的方法 getDateTImeInstance）并将其存储到 formatter 中。DateFormat 允许开发人员将日期和时间格式化成一个 String。第 422 行至第 423 行是将一个由当前系统日期、时间、三个空格以及 message 中的值所组成的 String 追加到 accessLogJTextArea 之中。第 422 行使用的是一个 new Date()，它将返回当前的日期和时间。然后，再利用 formatter 的 format 方法将其转换为一个经格式化后的 String 并通过 append 方法将其追加至 JTextArea 中。

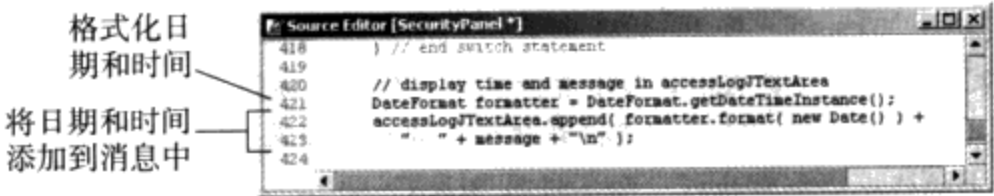


图 11.14 输出当前的日期、时间以及消息

7. 保存应用程序 保存修改后的源代码文件。

到现在为止，我们已定义了enterJButtonActionPerformed方法，接下来，将为每一个数字JButton和C JButton创建各自所应有的方法。

编写剩余方法

- 1. 编写 oneJButtonActionPerformed 方法 将图 11.16 中第 313 行至第 314 行插入到此方法中。这些代码行将通过使用+运算符把字符串值"1"追加到 securityCodeJPasswordField 的 password 属性值的后面。这样做的目的是希望能够将数字 JButton 的值追加到当前 JPasswordField 中访问码的后面。

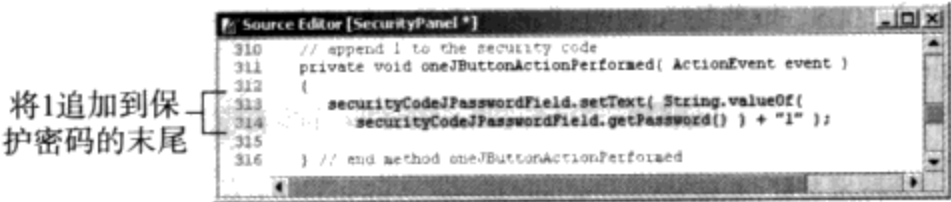


图 11.15 将 1 追加至保护密码的末尾

- 2. 编写其余数字 JButton 的方法 针对其余的数字 JButton (2~9, 以及 0), 重复步骤 1 中的操作。一定要将双引号中的值替换成相应 JButton 上的数字 (比如, twoJButtonActionPerformed 是将 "2" 追加至当前 securityCodeJPasswordField 中访问码的后面)。图 11.16 中显示了 twoJButton 和 threeJButton 这两个 JButton 所应有的方法。
- 3. 编写 clearJButtonActionPerformed 方法 加入图 11.17 中第 393 行。该行将用于清除 Security code: JPasswordField 中的内容。
- 4. 保存应用程序 保存修改后的源代码文件。
- 5. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\SecurityPanel 进入到当前的工作目录中。

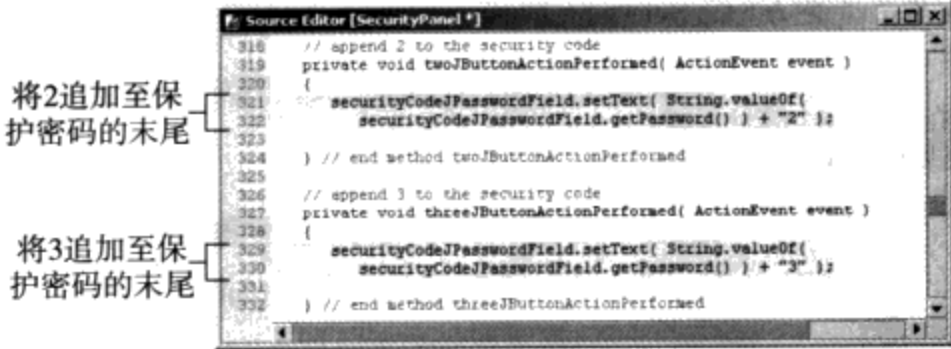


图 11.16 编写属于 2 JButton 和 3 JButton 的方法；其余 JButton 的方法类似

- 6. 编译应用程序 通过键入 javac SecurityPanel.java 编译该应用程序。
- 7. 运行应用程序 若此应用程序能正确编译，通过键入 java SecurityPanel 来运行它。图 11.18 中显示了完成后的应用程序的运行结果。利用小键盘输入以下的代码：9998，8345，7777，8，1006 和 8345。切记，在每一个代码的后面，应按下 # 号键。
- 8. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- 9. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

清理保护密码中的内容

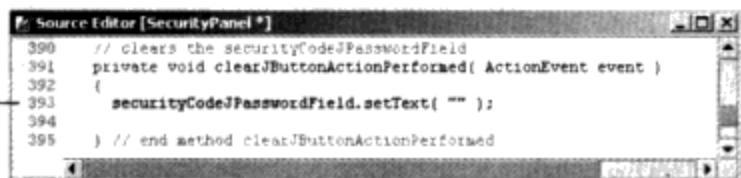


图 11.17 清理 Security Code:JPasswordField

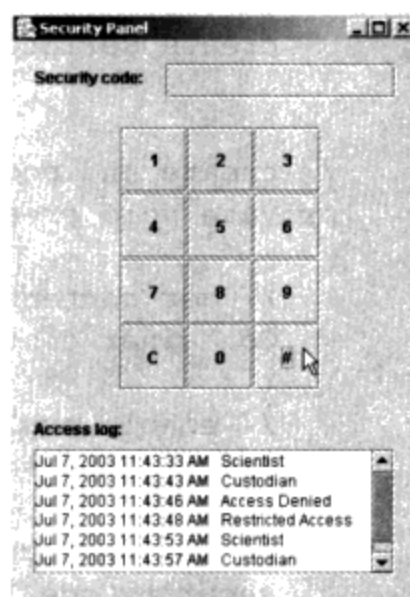


图 11.18 完成后的门禁系统应用程序

图 11.19 中给出了门禁系统应用程序的完整源代码。本教程中，凡需要添加、查看或是修改的代码，均在图中相应的代码行中做了突出显示。

```

1 // Tutorial 11: SecurityPanel.java
2 // Enable user to enter security codes specifying access privileges.
3 import java.awt.*;
4 import java.awt.event.*;
5 import java.text.DateFormat;
6 import java.util.Date;
7 import javax.swing.*;
8
9 public class SecurityPanel extends JFrame
10 {
11     // JLabel and JPasswordField for user to input security code
12     private JLabel securityCodeJLabel;
13     private JPasswordField securityCodeJPasswordField;
14
15     // JButtons to represent security keypad
16     private JButton oneJButton;
17     private JButton twoJButton;
18     private JButton threeJButton;
19     private JButton fourJButton;
20     private JButton fiveJButton;
21     private JButton sixJButton;
22     private JButton sevenJButton;
23     private JButton eightJButton;
24     private JButton nineJButton;
25     private JButton clearJButton;
26     private JButton zeroJButton;
27     private JButton enterJButton;
28
29     // JLabel, JTextArea and JScrollPane to display access log
30     private JLabel accessLogJLabel;
31     private JTextArea accessLogJTextArea;
32     private JScrollPane accessLogJScrollPane;
33
34     // no-argument constructor
35     public SecurityPanel()
36     {

```

```
37     createUserInterface();
38 }
39
40 // create and position GUI components; register event handlers
41 private void createUserInterface()
42 {
43     // get content pane for attaching GUI components
44     Container contentPane = getContentPane();
45
46     // enable explicit positioning of GUI components
47     contentPane.setLayout( null );
48
49     // set up securityCodeJLabel
50     securityCodeJLabel = new JLabel();
51     securityCodeJLabel.setBounds( 16 , 16 , 90 , 21 );
52     securityCodeJLabel.setText( "Security code:" );
53     contentPane.add( securityCodeJLabel );
54
55     // set up securityCodeJPasswordField
56     securityCodeJPasswordField = new JPasswordField();
57     securityCodeJPasswordField.setBounds( 114, 16 , 172, 26 ); 设置JPasswordField
58     securityCodeJPasswordField.setEditable( false ); 的bounds属性和editable属性
59     contentPane.add( securityCodeJPasswordField );
60
61     // set up oneJButton
62     oneJButton = new JButton();
63     oneJButton.setBounds( 80 , 64 , 50 , 50 );
64     oneJButton.setText( "1" );
65     contentPane.add( oneJButton );
66     oneJButton.addActionListener(
67
68         new ActionListener() // anonymous inner class
69         {
70             // event handler called when oneJButton is pressed
71             public void actionPerformed((ActionEvent event)
72             {
73                 oneJButtonActionPerformed( event );
74             }
75
76         } // end anonymous inner class
77
78     ); // end call to addActionListener
79
80     // set up twoJButton
81     twoJButton = new JButton();
82     twoJButton.setBounds( 130, 64 , 50 , 50 );
83     twoJButton.setText( "2" );
84     contentPane.add( twoJButton );
85     twoJButton.addActionListener(
86
87         new ActionListener() // anonymous inner class
88         {
89             // event handler called when twoJButton is pressed
90             public void actionPerformed((ActionEvent event)
91             {
```

```
92         twoJButtonActionPerformed( event );
93     }
94
95     } // end anonymous inner class
96
97     ); // end call to addActionListener
98
99     // set up threeJButton
100    threeJButton = new JButton();
101    threeJButton.setBounds( 180 , 64 , 50 , 50 );
102    threeJButton.setText( "3" );
103    contentPane.add( threeJButton );
104    threeJButton.addActionListener(
105
106        new ActionListener() // anonymous inner class
107        {
108            // event handler called when threeJButton is pressed
109            public void actionPerformed((ActionEvent event) )
110            {
111                threeJButtonActionPerformed( event );
112            }
113
114        } // end anonymous inner class
115
116    ); // end call to addActionListener
117
118    // set up fourJButton
119    fourJButton = new JButton();
120    fourJButton.setBounds( 80 , 114, 50 , 50 );
121    fourJButton.setText( "4" );
122    contentPane.add( fourJButton );
123    fourJButton.addActionListener(
124
125        new ActionListener() // anonymous inner class
126        {
127            // event handler called when fourJButton is pressed
128            public void actionPerformed((ActionEvent event) )
129            {
130                fourJButtonActionPerformed( event );
131            }
132
133        } // end anonymous inner class
134
135    ); // end call to addActionListener
136
137    // set up fiveJButton
138    fiveJButton = new JButton();
139    fiveJButton.setBounds( 130, 114, 50 , 50 );
140    fiveJButton.setText( "5" );
141    contentPane.add( fiveJButton );
142    fiveJButton.addActionListener(
143
144        new ActionListener() // anonymous inner class
145        {
146            // event handler called when fiveJButton is pressed
```

```
147         public void actionPerformed((ActionEvent event)
148         {
149             fiveJButtonActionPerformed( event );
150         }
151
152     } // end anonymous inner class
153
154 ); // end call to addActionListener
155
156 // set up sixJButton
157 sixJButton = new JButton();
158 sixJButton.setBounds( 180 , 114, 50 , 50 );
159 sixJButton.setText( "6" );
160 contentPane.add( sixJButton );
161 sixJButton.addActionListener(
162
163     new ActionListener() // anonymous inner class
164     {
165         // event handler called when sixJButton is pressed
166         public void actionPerformed((ActionEvent event)
167         {
168             sixJButtonActionPerformed( event );
169         }
170
171     } // end anonymous inner class
172
173 ); // end call to addActionListener
174
175 // set up sevenJButton
176 sevenJButton = new JButton();
177 sevenJButton.setBounds( 80 , 164 , 50 , 50 );
178 sevenJButton.setText( "7" );
179 contentPane.add( sevenJButton );
180 sevenJButton.addActionListener(
181
182     new ActionListener() // anonymous inner class
183     {
184         // event handler called when sevenJButton is pressed
185         public void actionPerformed((ActionEvent event)
186         {
187             sevenJButtonActionPerformed( event );
188         }
189
190     } // end anonymous inner class
191
192 ); // end call to addActionListener
193
194 // set up eightJButton
195 eightJButton = new JButton();
196 eightJButton.setBounds( 130, 164 , 50 , 50 );
197 eightJButton.setText( "8" );
198 contentPane.add( eightJButton );
199 eightJButton.addActionListener(
200
201     new ActionListener() // anonymous inner class
```



```
202     {
203         // event handler called when eightJButton is pressed
204         public void actionPerformed((ActionEvent event) )
205         {
206             eightJButtonActionPerformed( event );
207         }
208
209     } // end anonymous inner class
210
211 ); // end call to addActionListener
212
213 // set up nineJButton
214 nineJButton = new JButton();
215 nineJButton.setBounds( 180 , 164 , 50 , 50 );
216 nineJButton.setText( "9" );
217 contentPane.add( nineJButton );
218 nineJButton.addActionListener(
219
220     new ActionListener() // anonymous inner class
221     {
222         // event handler called when nineJButton is pressed
223         public void actionPerformed((ActionEvent event) )
224         {
225             nineJButtonActionPerformed( event );
226         }
227
228     } // end anonymous inner class
229
230 ); // end call to addActionListener
231
232 // set up clearJButton
233 clearJButton = new JButton();
234 clearJButton.setBounds( 80 , 214, 50 , 50 );
235 clearJButton.setText( "C" );
236 contentPane.add( clearJButton );
237 clearJButton.addActionListener(
238
239     new ActionListener() // anonymous inner class
240     {
241         // event handler called when clearJButton is pressed
242         public void actionPerformed((ActionEvent event) )
243         {
244             clearJButtonActionPerformed( event );
245         }
246
247     } // end anonymous inner class
248
249 ); // end call to addActionListener
250
251 // set up zeroJButton
252 zeroJButton = new JButton();
253 zeroJButton.setBounds( 130, 214, 50 , 50 );
254 zeroJButton.setText( "0" );
255 contentPane.add( zeroJButton );
256 zeroJButton.addActionListener(
```

```
257
258     new ActionListener() // anonymous inner class
259     {
260         // event handler called when zeroJButton is pressed
261         public void actionPerformed((ActionEvent event) )
262         {
263             zeroJButtonActionPerformed( event );
264         }
265     } // end anonymous inner class
266
267 ); // end call to addActionListener
268
269 // set up enterJButton
270 enterJButton = new JButton();
271 enterJButton.setBounds( 180 , 214, 50 , 50 );
272 enterJButton.setText( "#" );
273 contentPane.add( enterJButton );
274 enterJButton.addActionListener(
275
276     new ActionListener() // anonymous inner class
277     {
278         // event handler called when enterJButton is pressed
279         public void actionPerformed((ActionEvent event) )
280         {
281             enterJButtonActionPerformed( event );
282         }
283     } // end anonymous inner class
284
285 ); // end call to addActionListener
286
287 // set up accessLogJLabel
288 accessLogJLabel = new JLabel();
289 accessLogJLabel.setBounds( 16 , 285, 100, 16 );
290 accessLogJLabel.setText( "Access log:" );
291 contentPane.add( accessLogJLabel );
292
293 // set up accessLogJTextArea
294 accessLogJTextArea = new JTextArea();
295
296 // set up accessLogJScrollPane
297 accessLogJScrollPane = new JScrollPane( accessLogJTextArea );
298 accessLogJScrollPane.setBounds( 16 , 309 , 270, 95 );
299 contentPane.add( accessLogJScrollPane );
300
301 // set properties of application's window
302 setTitle( "Security Panel" ); // set title bar string
303 setSize( 310 , 450 ); // set window's size
304 setVisible( true ); // display window
305
306 } // end method createUserInterface
307
308 // append 1 to the security code
309 private void oneJButtonActionPerformed((ActionEvent event) )
310 {
```

```

313 securityCodeJPasswordField.setText( String.valueOf(
314 securityCodeJPasswordField.getPassword() ) + "1" );
315
316 } // end method oneJButtonActionPerformed
317
318 // append 2 to the security code
319 private void twoJButtonActionPerformed( ActionEvent event )
320 {
321 securityCodeJPasswordField.setText( String.valueOf(
322 securityCodeJPasswordField.getPassword() ) + "2" );
323
324 } // end method twoJButtonActionPerformed
325
326 // append 3 to the security code
327 private void threeJButtonActionPerformed( ActionEvent event )
328 {
329 securityCodeJPasswordField.setText( String.valueOf(
330 securityCodeJPasswordField.getPassword() ) + "3" );
331
332 } // end method threeJButtonActionPerformed
333
334 // append 4 to the security code
335 private void fourJButtonActionPerformed( ActionEvent event )
336 {
337 securityCodeJPasswordField.setText( String.valueOf(
338 securityCodeJPasswordField.getPassword() ) + "4" );
339
340 } // end method fourJButtonActionPerformed
341
342 // append 5 to the security code
343 private void fiveJButtonActionPerformed( ActionEvent event )
344 {
345 securityCodeJPasswordField.setText( String.valueOf(
346 securityCodeJPasswordField.getPassword() ) + "5" );
347
348 } // end method fiveJButtonActionPerformed
349
350 // append 6 to the security code
351 private void sixJButtonActionPerformed( ActionEvent event )
352 {
353 securityCodeJPasswordField.setText( String.valueOf(
354 securityCodeJPasswordField.getPassword() ) + "6" );
355
356 } // end method sixJButtonActionPerformed
357
358 // append 7 to the security code
359 private void sevenJButtonActionPerformed( ActionEvent event )
360 {
361 securityCodeJPasswordField.setText( String.valueOf(
362 securityCodeJPasswordField.getPassword() ) + "7" );
363
364 } // end method sevenJButtonActionPerformed
365
366 // append 8 to the security code
367 private void eightJButtonActionPerformed( ActionEvent event )
368 {

```

将数字 JButton 的值 "1" 追加至 JPasswordField 中所存储的密码中

将数字 JButton 的值 "2" 追加至 JPasswordField 中所存储的密码中

将数字 JButton 的值 "3" 追加至 JPasswordField 中所存储的密码中

将数字 JButton 的值 "4" 追加至 JPasswordField 中所存储的密码中

将数字 JButton 的值 "5" 追加至 JPasswordField 中所存储的密码中

将数字 JButton 的值 "6" 追加至 JPasswordField 中所存储的密码中

将数字 JButton 的值 "7" 追加至 JPasswordField 中所存储的密码中

```

369 securityCodeJPasswordField.setText( String.valueOf(
370 securityCodeJPasswordField.getPassword() ) + "8" );
371
372 } // end method eightJButtonActionPerformed
373
374 // append 9 to the security code
375 private void nineJButtonActionPerformed((ActionEvent event) )
376 {
377 securityCodeJPasswordField.setText( String.valueOf(
378 securityCodeJPasswordField.getPassword() ) + "9" );
379
380 } // end method nineJButtonActionPerformed
381
382 // append 0 to the security code
383 private void zeroJButtonActionPerformed((ActionEvent event) )
384 {
385 securityCodeJPasswordField.setText( String.valueOf(
386 securityCodeJPasswordField.getPassword() ) + "0" );
387
388 } // end method zeroJButtonActionPerformed
389
390 // clears the securityCodeJPasswordField
391 private void clearJButtonActionPerformed((ActionEvent event) )
392 {
393 securityCodeJPasswordField.setText( "" );
394
395 } // end method clearJButtonActionPerformed
396
397 // gets access code and determines level of clearance
398 private void enterJButtonActionPerformed((ActionEvent event) )
399 {
400 String message; // displays access status of users
401
402 // stores access code entered
403 int accessCode = Integer.parseInt( String.valueOf(
404 securityCodeJPasswordField.getPassword() ) );
405
406 securityCodeJPasswordField.setText( "" );
407
408 switch ( accessCode ) // check access code input
409 {
410 // access code is 7, 8 or 9
411 case 7:
412 case 8:
413 case 9:
414 message = "Restricted Access" ;
415 break ; // done processing case
416
417 // access code equal to 1645
418 case 1645:
419 message = "Technician";
420 break; // done processing case
421
422 // access code equal to 8345

```

将数字 JButton
的值 "8" 追加至
JPasswordField 中
所存储的密码中

将数字 JButton
的值 "9" 追加至
JPasswordField 中
所存储的密码中

将数字 JButton
的值 "0" 追加至
JPasswordField 中
所存储的密码中

清除 JPasswordField 中的内容

显示消息的 String

取得 password 属性

清除 JPasswordField 中的内容

利用 switch 语句确定用户访问级别

可导致相同 message 的三个 case 标签

终止 switch 语句后执行 switch 后的下一条语句

为技术员提供的访问码 1645


```

423         case 8345:
424             message = "Custodian";           为管理员提供的访问码 8346
425             break; // done processing case
426
427         // access code equal to 9998 or between 1006 and 1008
428         case 9998:
429         case 1006:
430         case 1007:           为科学家提供的访问码
431         case 1008:           9998, 1006, 1007 和 1008
432             message = "Scientist" ;
433             break; // done processing case
434
435         // if no other case is true
436         default:             无 case 匹配时开始执行的 default case
437             message = "Access Denied";
438
439     } // end switch statement      结束 switch 语句的右花括号
440
441     // display time and message in accessLogJTextArea
442     DateFormat formatter = DateFormat.getDateTimeInstance();      将当前日期
443     accessLogJTextArea.append( formatter.format( new Date() ) +    和时间追加
444         " " + message + "\n" );      至 message 中
445
446 } // end method enterJButtonActionPerformed
447
448 // main method
449 public static void main( String[] args )
450 {
451     SecurityPanel application = new SecurityPanel();
452     application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
453
454 } // end method main
455
456 } // end class SecurityPanel

```

图 11.19 门禁系统应用程序的完整源代码

自测题

- 在默认情况下，当用户通过 JPasswordField 进行输入时，屏幕中将显示 ____。
 - 什么也没有
 - 星号
 - 所输入的文本
 - 其中仅有的数字
- ____ 将导致程序控制进入位于 switch 后的首条语句。
 - 分号
 - stop 关键字
 - break 语句
 - case 语句的结束

答案：1) b 2) c

11.4 小结

在本教程中，我们学习了如何使用 switch 多向选择语句，并探索了它与嵌套的 if...else 语句之间的相似性。我们还研究了一个可用于表明 switch 语句控制流程的 UML 活动图。

接着，利用所学到的知识创建了一个自己的门禁系统应用程序。通过使用这条 switch 语句，便可确定出用户是否输入了一个正确的保护密码。同时，还声明了一些 case 语句并包含了一个可作为

选用的 default case，它是在未提供有效的安检码时开始执行的。我们了解到，如果没有提供这个可选用的 default case，则相应的无效代码将会导致整个的 switch 语句被跳过，因此，提供 default case 是非常重要的。通过学习，还了解到，break 语句可结束 case 语句。如果一条 case 语句中缺省了 break 语句，那么，执行将“下落至”紧接着的下一个 case 语句。

在下一个教程中，将学习如何利用一些被称之为方法的属于小型的、易于管理的可复用代码片断，创建出所需要的应用程序。其实，我们一直都在使用现有的方法，不过，到那时，将教会读者如何去编写属于自己的方法——它们通常也被称为程序员定义的方法。之后，还将利用一些方法改进本书前面曾经创建过得一个工资计算器应用程序。

技术小结

编写一条 switch 语句

- 在关键字 switch 的后面跟上一个控制表达式。
- 为每一个 case 使用关键字 case，其后跟一个可与控制表达式进行比较的表达式。
- 对于每一个 case，声明用以执行与控制表达式相匹配的语句。
- 使用 break 语句结束每一条 case 语句。如果缺省了 break 语句，则执行将“下落至”下一条 case 语句。
- 如果控制表达式不能与所提供的任意一个 case 相匹配，那么，应在 default 标签的后面跟上需要执行的语句。

通过 JPasswordField 隐藏用户的输入

- 使用 JPasswordField 隐藏用户的输入。
- 通过方法 getPassword 取得用户在 JPasswordField 中的输入。
- (选用) 利用方法 setEchoChar 指定 JPasswordField 中所显示的回应字符。

取得一个包含当前日期和时间的格式化 String

- 利用 DateFormat.getDateTimeInstance 取得一个用以显示日期和时间的 DateFormat。
- 将 new Date() 传递给 DateFormat 的 format 方法，可获得包含格式化的日期和时间的 String。

关键术语

break 语句 通常出现在每个 case 的末尾处。该语句用于立即终止 switch 语句，使程序控制进入到紧接着 switch 语句后的下一条语句。

case 标签 位于所执行的语句之前，用于执行的语句是在 switch 的控制表达式同该 case 标签中的表达式相匹配时开始执行的。

char 类型 用于存储字符值的一种类型。

字符常量 字符字面值的另一种名称。

字符面值 一个属于 char 类型的变量值，它是由位于一对单引号之间的一个字符来表示的，比如，'A'，'d'，'*'，' ' 等。

常量表达式 一个不能够被更改的值。case 标签是由关键字 case 后跟一个常量表达式所组成的。该常量表达式必须是一个字符面值或者是一个整数值。

控制表达式 switch 语句中的一个表达式，该表达式的值将按照从上到下的顺序同每一个 case 进行比较直至找到一个与之相匹配的 case 并执行它，或者是执行 default case，或者是到达右花括号。

Date 类 一个用于存储日期和时间信息的类。

DateFormat 类 用于格式化日期和时间信息的类。

default case 一个可选用的 case，其中的语句是在 switch 的控制表达式不能够与任意一个 case 值匹配时开始执行的。

回应字符 用于替换 JPasswordField 中所显示的每一个字符。默认的回应字符是 *，但程序员可通过调用 JPassword 的 setEchoChar 方法来指定所需要的回应字符。

JPasswordField 的 editable 属性 确定是否允许用户在 JPasswordField 中进行输入。

getPassword 方法 返回 JPasswordField 中的文本。

JPasswordField 显示或输入一个密码。在 JPasswordField 中所显示的字符将被替换成星号 (*), 并且能够允许用户向可编辑的 JPasswordField 中输入密码。

屏蔽字符 回应字符的另一种名称。

多向选择语句 一种语句, 如 switch 语句, 可根据控制表达式的值在多个操作 (或操作序列) 之间选择一个来执行。

JPasswordField 的 password 属性 存储 JPasswordField 中所输入的文本。

setEchoChar 方法 用于指定 JPasswordField 中的回应字符。

switch 语句 通过将控制表达式与一系列 case 值进行比较从而做出选择的多向选择语句。匹配完以后, 会根据这些值采取不同的操作。

GUI 设计导航

总体设计

- 如果进行开发的 GUI 被用于模拟现实世界中的某个物体的话, 则相应的 GUI 设计就应该模拟出该物体的物理外观。

JPasswordField

- 利用 JPasswordField 屏蔽密码或其他敏感信息。

Java 类库索引

Date 代表日期和时间。通过编写 new Date() 取得当前的系统日期和时间。

DateFormat 一个用于格式化 Date 对象的类。

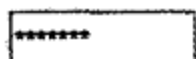
● 方法

format 格式化一个 Date 对象并返回一个 String。

getDateTimeInstance 返回一个用于格式化 Date 对象的 DateFormat 对象。

JPasswordField 允许用户通过键盘输入数据, 并出于安全的目的将每一个字符回显为一个星号 (*)。

● 运行



● 方法

setBounds 通过设置 bounds 属性确定某个 JPasswordField 的位置与大小。

setEditable 指定用户是否可对 JPasswordField 进行编辑。

getPassword 返回 JPasswordField 中的密码。

setEchoChar 指定 JPasswordField 的回应字符。

setText 指定 JPasswordField 中应显示的文本。

习题

选择题

- 11.1 _____ 表示一条 switch 语句的结束。
a) end 关键字 b) } 字符 c) break 关键字 d) default 关键字
- 11.2 表达式 _____ 将返回当前的系统时间和日期。
a) DateFormat.getDateTime() b) new Date()
c) DateFormat.getDateTimeInstance() d) new CurrentDate()
- 11.3 利用一个 _____ 组件可隐藏用户输入的信息; 默认情况下, 用户输入的每一个字符将被回显为一个星号 (*)。

- d) **确定售货员的佣金百分比** 在 `calculateJButtonActionPerformed` 中第 131 行, 声明一个用于存储佣金百分比的 `int` 型局部变量 `commission`。接下来, 插入一条 `switch` 语句, 通过所售商品的数量确定出某售货员应有的佣金百分比。在这条 `switch` 语句中, 将佣金百分比作为一个整数赋值给变量 `commission`。例如, 若佣金百分比为 2%, 则将 2 赋值给 `commission`。控制表达式应为 `items/10`。由于这是一个整数间的算术运算, 任何处于范围 0~9 之间的商品数量其计算的结果都将为 0, 而任何处于范围 10~19 之间的商品数量其计算的结果都将为 1, 等等。在 `switch` 语句内, 提供相应的能使该 `switch` 测试出问题陈述中所指定范围值的 `case`。
 - e) **计算佣金百分比的小数值** 在 `switch` 语句的后面, 插入一条语句, 将 `commission` 除以 100.0, 并将结果赋值给 `double` 型的局部变量 `commissionRate`。
 - f) **计算售货员的收入所得** 在步骤(e)中所插入语句的后面, 再插入一条语句, 将售货员的 `sales` [已由步骤(c)计算出] 乘以 `commissionRate`, 然后将结果赋值给 `double` 型的局部变量 `earnings`。
 - g) **显示销售总额、佣金百分比以及售货员的收入所得** 在方法 `calculateJButtonActionPerformed` 中所创建的 `DecimalFormat dollars` 语句的后面, 添加三条用于将局部变量 `sales`, `commission` 和 `earnings` 的值分别显示在 `salesJTextField`, `commissionJTextField` 和 `earningsJTextField` 中的语句。对于销售总额和收入值, 利用 `DecimalFormat` 的 `dollars`, 将这些值格式化为美元的形式。
 - h) **保存应用程序** 保存修改后的源代码文件。
 - i) **打开命令提示符窗口改变目录** 选择 `Start` → `Programs` → `Accessories` → `Command Prompt` 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\SalesCommissionCalculator` 进入到当前的工作目录中。
 - j) **编译应用程序** 通过键入 `javac SalesCommissionCalculator.java`, 编译该应用程序。
 - k) **运行完成后的应用程序** 若应用程序能正确编译, 通过键入 `java SalesCommissionCalculator` 来运行它。为测试应用程序, 在 `Number of items sold:JTextField` 中输入一些不同的数, 然后点击 `Calculate JButton`, 确定是否正确计算出了所销售的佣金额。
 - l) **关闭应用程序** 点击关闭按钮关闭正在运行的应用程序。
 - m) **关闭命令提示符窗口** 点击命令提示符窗口的关闭按钮将其关闭。
- 11.12 (联邦所得税计算器应用程序) 创建一个应用程序, 计算个人需缴纳的联邦所得税 [提示: 事实上, 美国联邦所得税会因多种因素而变化。如果需要了解更多的信息, 请到 www.irs.gov/pub/irs-pdf/f1040e03.pdf 上查看美国国税局 (IRS: Internal Revenue Service) 1040-ES 表中的相关内容]。该应用程序的操作结果如图 11.21 所示。假定所输入的收入都是大于或等于 0 的整数。试根据下列的收入范围及相应的税率, 对个人所得税进行计算:

低于 25 000 美元 = 2% 所得税
25 000~49 999 美元 = 5% 所得税
50 000~74 999 美元 = 10% 所得税
75 000~99 999 美元 = 15% 所得税
100 000 美元及以上 = 20% 所得税

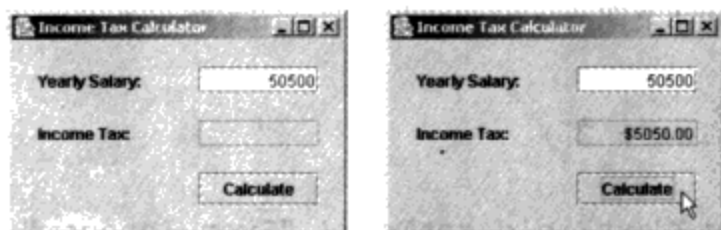


图 11.21 联邦所得税计算器 GUI

- a) **将模板复制到工作目录中** 将 `C:\Examples\Tutorial11\Exercises\IncomeTaxCalculator` 目录复制到 `C:\SimplyJava` 目录中。
- b) **打开模板文件** 在自己的文本编辑器中打开 `IncomeTaxCalculator.java` 文件。
- c) **在 `calculateJButtonActionPerformed` 方法中插入一条 `switch` 语句** 在 `calculateJButtonActionPerformed` 方法内 (位于模板第 110 行至第 118 行), 将一条 `switch` 语句插入到第 114 行, 所插入的 `switch` 语句将用于确定税率, 并将最终所得到的税率值赋予变量 `taxRate` (在第 112 行声明)。 `int` 型的变量 `salary` (第 113 行) 用做存储用户输入的收入。通过使用控制表达式 `salary/`

25000 来确定个人的税率大小。若收入小于 25 000 美元,则控制表达式的值为 0。对于范围在 25 000~49 999 美元之间的收入,控制表达式的值为 1。对于范围在 50 000~74 999 美元之间的收入,控制表达式的值为 2。对于范围在 75 000~99 999 美元之间的收入,控制表达式的值为 3。对于其余范围之间的收入,则使用 default case。

- d) 在 `calculateJButtonActionPerformed` 方法中计算并显示所得税 在步骤(c)中所插入的 switch 语句的后面,再插入一条语句,通过将 salary 乘以 taxRate 的值计算出个人的所得税,然后,把结果存储到 double 型的变量 incomeTax 中。在所创建的被称作 dollars 的 DecimalFormat 语句的后面,插入一条用于格式化 incomeTax 值的语句,并在 incomeTaxJTextField 中显示出经过格式化后的值。
- e) 保存应用程序 保存修改后的源代码文件。
- f) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\IncomeTaxCalculator` 进入到当前的工作目录中。
- g) 编译应用程序 通过键入 `javac IncomeTaxCalculator.java`,编译该应用程序。
- h) 运行完成后的应用程序 若应用程序能正确编译,通过键入 `java IncomeTaxCalculator` 来运行它。为测试应用程序,输入多个不同的年度收入,然后确认所计算出的所得税是否正确。
- i) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- j) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

11.13 (说出这段代码的作用)下列代码的运行结果是什么?可针对 grade 值分别为 'A', 'B', 'C', 'D', 'F' 及无效值时,确定出其相应的输出结果。

```

1 switch ( grade )
2 {
3     case 'A':
4         displayJLabel.setText( "Excellent!" );
5
6     case 'B':
7         displayJLabel.setText( "Very good!" );
8
9     case 'C':
10        displayJLabel.setText( "Good." );
11
12    case 'D':
13        displayJLabel.setText( "Poor." );
14
15    case 'F':
16        displayJLabel.setText( "Failure." );
17
18    default:
19        displayJLabel.setText( "Invalid grade." );
20 }

```

11.14 (找出代码中的错误)下面的这个 switch 语句将用于确定某个 int 是一个奇数还是一个偶数。请找出代码中的错误:

```

1 switch ( value % 2 )
2 {
3     case 0:
4         outputJTextField.setText( "Even Integer" );
5
6     case 1:
7         outputJTextField.setText( "Odd Integer" );
8         break;
9
10 } // end switch

```

11.15 (使用调试程序)(折扣计算器应用程序)折扣计算器应用程序将根据用户的消费额大小,确定该用户应得的折扣额。对于购买额在 150 美元或超过 150 美元的顾客,可享受 15% 的折扣;对于购买额在 100~149 美元之间的顾客,可享受 10% 的折扣;对于购买额在 50~99 美元之间的顾客,可享受 5% 的折扣;而小于 50 美元购买额的顾客将不能享受到折扣的优惠。在测试应用程序的

过程中, 会注意到该应用程序不能为某些值计算出正确的折扣。通过使用调试程序查找并改正该应用程序中的逻辑错误。图 11.22 显示的是针对每一个范围内的值所应得到的正确结果。

- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial11\Exercises\Debugger\DiscountCalculator 目录复制到 C:\SimplyJava 目录中。

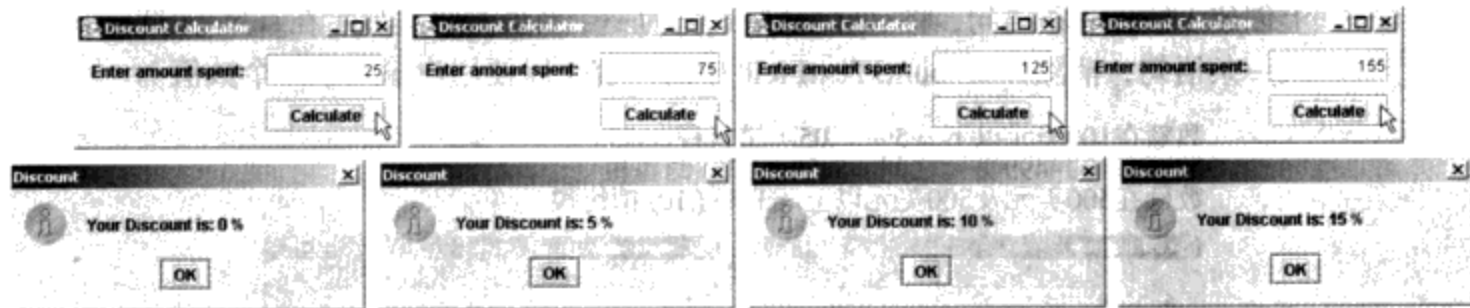


图 11.22 折扣计算器应用程序的正确输出结果

- b) 打开模板文件 在自己的文本编辑器中打开 DiscountCalculator.java 文件。
- c) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\DiscountCalculator` 进入到当前的工作目录中。
- d) 因调试需要编译应用程序 通过键入 `javac -g DiscountCalculator.java`, 编译该应用程序。
- e) 运行应用程序 若应用程序能正确编译, 通过键入 `java DiscountCalculator` 来运行它。为测试应用程序, 输入图 11.22 中所显示的数值。当输入值 75 (或者是处于范围 50~99 美元之间的其他值) 时, 注意应用程序错误地指示出了 15% 的折扣。
- f) 启动调试程序 关闭应用程序 (使命令提示符窗口仍处于打开状态), 然后通过输入 `jdb` 启动调试程序。
- g) 寻找并更正错误 利用前面教程中学到的调试技巧, 确定应用程序中的逻辑错误。在 `discount-JButtonActionPerformed` 方法中每一个 case 内的 `break` 语句 (位于第 81 行和第 85 行), 以及第 93 行 (`switch` 之后的第一条语句) 上分别设置出相应的断点。每当应用程序进入中止模式时, 利用调试程序中的 `print` 命令检查变量 `discountRate` 的值。当检查完每个断点处的 `discountRate` 以后, 通过调试程序中的 `cont` 命令使应用程序能够继续执行。一旦发现逻辑错误, 应立即修改该应用程序中的 `switch` 语句, 使之能够选择出一个正确的折扣率。
- h) 保存应用程序 保存修改后的源代码文件。
- i) 编译应用程序 通过键入 `javac DiscountCalculator.java`, 编译该应用程序。
- j) 运行完成后的应用程序 若此应用程序能正确编译, 键入 `java DiscountCalculator` 来运行它。利用图 11.22 中所显示的数值对该应用程序进行测试, 确保能够得到正确的折扣额。
- k) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- l) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

挑战题

11.16 (收银机应用程序) 创建一个收银机应用程序 (参见图 11.23), 其上有一个小键盘, 类似于本教程中所构建的门禁系统应用程序上的小键盘。小键盘上除了一些数字键以外, 该收银机应用程序中还包含了一个作为小数点来使用的 `JButton`。同时, 还包含有几个 `Enter`, `Total`, `Delete` 和 `Clear` `JButton`。这些 `JButton` 用于分别向小计额中添加一个数值、计算税金及总额、在 `JTextField` 中删除当前的数值 (表示用户在输入过程中出现了差错) 以及清除当前所显示的数值。我们需要实现针对 `Enter` 和 `Total` `JButton` 功能的方法。假设用户输入的是一个非 0 正值。利用 `switch` 语句为所购买的商品数额计算出相应的销售税。之后, 通过将税金加入到小计额中计算出最终的总额大小, 并向用户显示出小计额、税金以及总额。使用一个 `switch` 语句, 根据以下的关系确定出相应的税率大小:

- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial11\Exercises\CashRegister 目录复制到 C:\SimplyJava 目录中。

- b) **打开模板文件** 在自己的文本编辑器中打开 CashRegister.java 文件。
- c) **在 enterJButtonActionPerformed 方法中添加代码** 在 enterJButtonActionPerformed 方法中(位于模板代码第 496 行至第 499 行), 插入一条语句, 将 amountJTextField 中的值转换为一个 double 类型并把该值累加到实例变量 subtotal 上。通过使用 DecimalFormat 的变量 dollars (已在第 56 行上创建) 插入一条语句, 将 subtotal 格式化为一个新值并将其显示在 subtotalJTextField 中。插入一条语句, 清除掉 amountJTextField 中的内容 (以使用户输入下一个数值)。

数额在 100 美元以下 = 5% (.05) 销售税
 数额在 100~499 美元之间 = 7.5% (.075) 销售税
 数额在 500 美元及 500 美元以上 = 10% (.10) 销售税

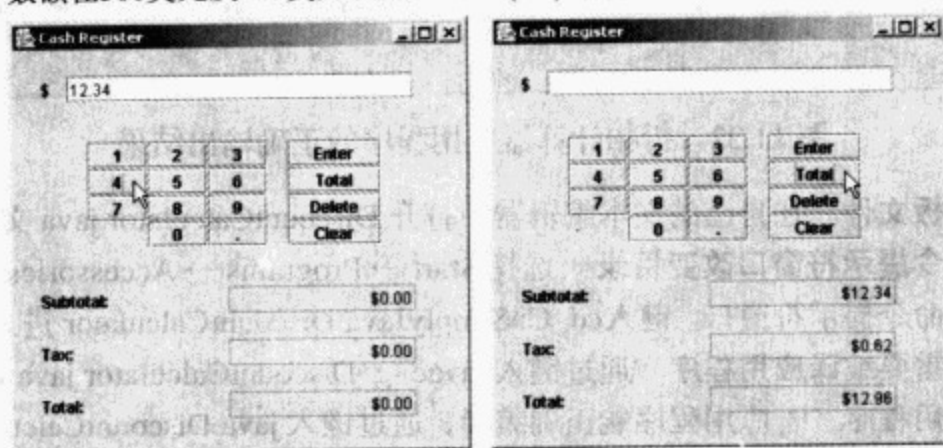


图 11.23 收银机应用程序 GUI

- d) **确定税率** 在 totalJButtonActionPerformed 方法中 (紧接着模板代码中 enterJButtonActionPerformed 方法之后), 开始声明一个被称为 taxRate 的 double 型局部变量, 该变量将用于存储税率并在小计金额计算中得到使用。接着, 插入一条确定税率的 switch 语句并把该值赋予变量 taxRate。在 switch 语句中, 利用控制表达式 (int) subtotal/100, 将小计金额转换成一个 int, 然后除以 100。该表达式的值在小计金额小于 100 美元时为 0, 在 100~499 美元范围之间时将分别为 1~4 之间的值。其余所有的值应通过本练习中的 default case 进行处理。
- e) **计算并显示税金及其总额** 在 totalJButtonActionPerformed 方法中 [位于步骤(d)中所插入的语句之后], 插入一条语句, 将 subtotal 乘以步骤(d)中已得到确定的 taxRate, 然后将结果存储到 double 型的变量 tax 中。接着, 将 subtotal 和 tax 相加并将结果存储到 double 型的变量 total 中。利用 DecimalFormat 的变量 dollars (已在第 56 行上创建), 插入一些语句, 格式化 tax 和 total 的值并把格式化后的值分别显示在 taxJTextField 和 totalJTextField 中。
- f) **清理 amountJTextField 中的内容并复位小计金额** 在 totalJButtonActionPerformed 方法中 [位于步骤(e)中所插入的语句之后], 插入一些语句, 清除 amountJTextField 中的内容并将 subtotal 复位至 0.0。
- g) **保存应用程序** 保存修改后的源代码文件。
- h) **打开命令提示符窗口改变目录** 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\CashRegister 进入到当前的工作目录中。
- i) **编译应用程序** 通过键入 javac CashRegister.java, 编译该应用程序。
- j) **运行完成后的应用程序** 若应用程序能正确编译, 通过键入 java CashRegister 来运行它。为测试应用程序, 利用数字 JButton 和 Enter JButton 输入多个不同的消费额, 通过点击 Total JButton 确定所得的税金及其总额是否正确。同时, 还应测试 Delete 和 Clear JButton 以确保对 JTextField 中相应的数字进行删除以及清除 JTextField 中内容的操作均能够得到正确地执行。
- k) **关闭应用程序** 点击关闭按钮关闭正在运行的应用程序。
- l) **关闭命令提示符窗口** 点击命令提示符窗口的关闭按钮将其关闭。



教程 12 改进的工资额计算器应用程序

引入方法的概念

教学目标

在本教程中，读者将学到以下内容：

- 通过方法创建模块化应用程序
- 使用“内置”方法
- 创建自己的方法

大多数解决实际问题的应用程序，要比本书前面所给出的应用程序复杂得多。经验表明，开发及维护一个较大规模应用程序的最好方法是从一个较小、可控的部件中开始创建。因此，这种方法被称为分而治之技术。在这些可控的部件中，包括一些被称之为方法的程序语句块，它们简化了较大规模应用程序的设计、实现及维护过程。通过本教程，将学习如何使用Java类库中预先定义好的方法，以及如何编写属于自己的方法。

12.1 探试改进的工资额计算器应用程序

我们将通过使用方法，改进教程6中所创建的工资额计算器应用程序。这个改进后的应用程序必须满足下面的需求：

应用程序需求分析

回顾教程6中的问题陈述：某公司将按照雇员一周内工作的小时数以及每小时的工资额，计算该员工一周内的总收入。创建一个应用程序，能够取得所需要的信息，通过将员工每小时的工资额乘以其工作的小时数，从而计算出他的总收入。在该应用程序中，假定一个标准的工作周为40个小时。一周内任何工作超过40个小时的时间则被认为属于加班时间，并因此将赚取相当于原工资1.5倍的工资（即每小时工资额的1.5倍）。1.5倍工资的计算是通过将员工每小时的工资额乘以1.5，然后再将这一计算后的结果乘以加班的小时数而求得的。之后，该值会被加到员工正常工作40个小时的收入上，最终计算出他一周内的总收入。需要设计一个方法，计算并返回该员工的工资。

完成后的应用程序将与教程6中的应用程序拥有完全相同的功能，但它使用了一个能更好地组织程序代码的编程形式——方法。应用程序将按照雇员每小时的工资额及工作的小时数计算出他一周内的工资。正常情况下，当员工工作40个小时或者少于40个小时的时候，所赚取的工资额为每小时工资额乘以工作的小时数。只有当员工工作超过工作周内标准的40个小时的时候，其计算才是不同的。在本教程中，将学习用于执行不同运算的方法，这些方法的计算结果取决于应用程序所执行的路径，路径不同，则得到的结果也将不同。我们将以这个完成后的应用程序的探试作为开始。之后，学习另外一些Java技术以最终创建出一个属于自己的应用程序。



探试改进的工资额计算器应用程序

1. 定位到完成后的应用程序 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\Examples\Tutorial12\CompletedApplication\WageCalculator2` 将目录改变到这个完成后的工资额计算器应用程序的目录下面。
2. 运行工资额计算器应用程序 在命令提示符窗口下键入 `java WageCalculator` 来运行这个应用程序（如图 12.1 所示）。提供给用户的 `JTextField` 将用做输入一周内每小时的工资额及工作的小时数。一旦输入这些值，便可点击 `Calculate JButton`，此时，一周内工作的总收入将显示在 `Gross wages:JTextField` 中。
3. 输入员工每小时的工资额及员工工作的小时数 在 `Hourly wage:JTextField` 中输入 10，在 `Hours worked:JTextField` 中输入 45。
4. 计算员工的全部收入 点击 `Calculate JButton`。所得结果（475.00 美元）将显示在 `Gross wages:JTextField` 中（如图 12.1 所示）。

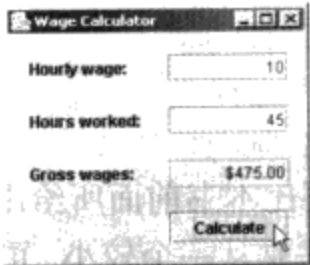


图 12.1 执行工资额计算器应用程序

5. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
6. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

12.2 类与方法

创建较大规模应用程序的关键是将应用程序分解成一些较小的部件。在面向对象的程序设计中，这些部件主要是由类所组成的，倘若再对类进一步细分，便可得到方法。

程序员通常是将其其他程序员定义类及方法同 Java 类库中现有的（也称为预定义的）可用代码结合在一起来使用。利用现有的代码，可节省开发的时间、精力和资金。这种代码复用的概念将提高程序开发人员开发应用程序的效率。图 12.2 中解释并列举了一些现有的 Java 方法。

方法	描述	举例
<code>Math.max(x, y)</code>	返回 x 和 y 之间的较大值	<code>Math.max(2.3, 12.7)</code> 的结果为 12.7 <code>Math.max(-2.3, -12.7)</code> 的结果为 -2.3
<code>Math.min(x, y)</code>	返回 x 和 y 之间的较小值	<code>Math.min(2.3, 12.7)</code> 的结果为 2.3 <code>Math.min(-2.3, -12.7)</code> 的结果为 -12.7
<code>Math.sqrt(x)</code>	返回 x 的平方根	<code>Math.sqrt(9)</code> 的结果为 3.0 <code>Math.sqrt(2)</code> 的结果为 1.4142135623731
<code>Integer.parseInt(x)</code>	将 String 转化为一个 int	<code>Integer.parseInt("1")</code> 的结果为 1
<code>Double.parseDouble(x)</code>	将 String 转化为一个 double	<code>Double.parseDouble("5.4")</code> 的结果为 5.4
<code>String.valueOf(x)</code>	返回 x 的 String 值	<code>String.valueOf(1.23)</code> 的结果为 "1.23"

图 12.2 一些预定义的 Java 方法

我们已经使用过一些 Java 类库中现有的类及其方法。例如，我们一直在应用程序中使用的 GUI 组件，实际上是 Java 类库中已预先定义好的类。与此同时，也使用了一些现有的 Java 类，如 `DecimalFormat`，利用该类可把应用程序的输出，格式化为一个适当的显示。如果没有 `DecimalFormat` 类，那么就需要自行来编写这一用于格式化的功能——完成这项任务，可能会需要很多行的代码，

而且还得使用一些这里仍未介绍过的编程技术。而这些现有类,已通过彻底的测试并去除了错误。在本书中,还将学习使用 Java 类库中更多的类及其方法。

当开发人员创建应用程序时,Java 中的类库并不一定能够提供出所有开发人员可以想像得到的功能,因而,Java 便允许开发人员创建属于自己的类和方法,从而满足特定应用程序的特殊需求。在下一节中,将学习如何定义方法;在随后的内容中,将创建属于自己的方法。在教程 18 中,还将学习如何创建自己的类。

自测题

1. _____ 为程序员提供了一些可完成通用任务的现有类。
a) Java 类库 b) preExisting 关键字 c) Java 代码库 d) library 关键字
2. 程序员通常会使用 _____。
a) 程序员定义的方法 b) 现有的方法
c) 程序员定义的方法以及现有的方法 d) 既不是程序员定义的方法也不是现有的方法

答案: 1) a 2) c

12.3 方法的定义

本书前面所给出的应用程序,均是通过调用 Java 类库中的方法(如 `Integer.parseInt`)完成应用程序应实现的任务。以下,将学习如何编写属于自己的程序员定义的方法。在创建改进的工资额计算器应用程序之前,首先学习如何在两个规模较小的应用程序中创建方法。之后,第一个应用程序是利用勾股定理计算直角三角形的斜边。第二个应用程序是计算三个数中的最大值。下面,让我们先回顾一下勾股定理。勾股定理指出:直角三角形(包含一个 90 度角的三角形)总是满足下面的关系——三角形中两个较小边边长的平方和,等于该三角形最大边(即斜边)边长的平方。在即将创建的这个应用程序中,两个较小边分别被称为边 A 和边 B,利用它们的长度计算斜边的长度。

创建斜边计算器应用程序

1. 将模板复制到工作目录中 将 `C:\Examples\Tutorial12\TemplateApplication\HypotenuseCalculator` 目录复制到 `C:\SimplyJava` 目录中。
2. 打开命令提示符窗口改变目录 选择 `Start → Programs → Accessories → Command Prompt` 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\HypotenuseCalculator` 进入到当前的工作目录中。
3. 运行斜边计算器模板应用程序 在命令提示符窗口中,通过键入 `java HypotenuseCalculator` 运行该应用程序(参见图 12.3)。当应用程序运行以后,在 `Length of side A:` 和 `Length of side B:` `JTextField` 中分别输入三角形两条较短边的边长,然后点击 `Calculate JButton`。这时,应用程序会计算出斜边的长度并将结果显示在输出框 `Length of hypotenuse:JTextField` 中。

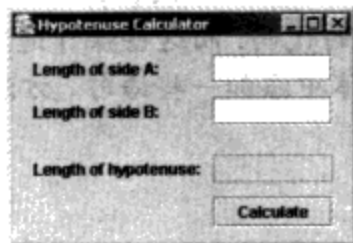


图 12.3 运行斜边计算器应用程序

4. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
5. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。
6. 打开斜边计算器应用程序的模板文件 在自己的文本编辑器中打开模板文件 `HypotenuseCalculator.java`。

7. 考察模板代码 考察图 12.4 中所显示的模板代码。这里，为 Calculate JButton 提供了一个尚未完善的方法（第 107 行至第 124 行）。在该方法中，声明了两个局部变量（第 111 行至第 114 行）。变量 sideA 和 sideB 中将分别用于存储已转换为 double 值的边 A 和边 B 的长度。第 117 行至第 122 行中含有一条 if 语句。该 if 语句的语句体会在所输入的边 A 和边 B 的长度为负值（或 0）时，显示出一个消息对话框。
8. 创建一个空的方法 将图 12.5 中的第 126 行至第 130 行添加到 calculateJButtonActionPerformed 方法的后面。注意第 130 行上有一个注释，该注释用于标识此方法已经结束。



好的编程习惯
在方法结尾处添加注释以表明该方法已经结束。

边A和边B的长度

当输入负值（或0）时会显示出一个消息对话框

```
Source Editor [HypotenuseCalculator *]
105 } // end method createUserInterface
106
107 // calculate and display hypotenuse length
108 private void calculateJButtonActionPerformed((ActionEvent event)
109 {
110     // get input from JTextFields
111     double sideA = Double.parseDouble(
112         lengthSideATextField.getText());
113     double sideB = Double.parseDouble(
114         lengthSideBTextField.getText());
115
116     // display error message if user enters invalid input
117     if (sideA <= 0 || sideB <= 0)
118     {
119         JOptionPane.showMessageDialog(null,
120             "You must enter positive numbers",
121             "Invalid Input Entered", JOptionPane.ERROR_MESSAGE);
122     }
123
124 } // end method calculateJButtonActionPerformed
125
```

图 12.4 斜边计算器的模板代码

方法头

用于划定方法定义结束的右花括号

```
Source Editor [HypotenuseCalculator *]
124 } // end method calculateJButtonActionPerformed
125
126 // return the square of side
127 private double square(double side)
128 {
129
130 } // end method square
131
```

图 12.5 定义方法 square

9. 理解方法 第 127 行上的这个方法起始于关键字 private 和 double，之后为一个方法名（本例中，方法名为 square）。方法名可以是任何有效的标识符。方法名的后面有一对含有一个变量声明的圆括号。在 private 和 square 之间的类型名（此时为 double）称为返回类型。方法可以为调用程序返回一条信息，或者根本不返回信息。如果方法不返回信息，它的返回类型需要使用关键字 void 进行声明。
- 位于圆括号对之间的声明称为参数列表，此处将声明一些变量（也称为参数）。尽管这个应用程序的参数列表中只包含了一个声明，而事实上，参数列表中可包含多个由逗号分隔的声明。参数列表中将声明每一个参数的类型和名称，所得到的这个参数变量将会在方法体中进行使用。通过使用参数，方法便可接收完成方法任务所需要的数据。
- 方法的第 1 行（包括像关键字 private 或 public，返回类型，方法名以及参数列表）被称为方法头。square 方法的方法头将声明一个 double 型参数变量 side 并将 square 的返回类型设置为 double。
- 在图 12.5 中第 128 行和第 130 行上的花括号之间，出现的所有声明和语句便构成了该方法的方法体。方法体中包含了用以执行某些操作的代码，而这些操作通常都需要对参数列表中的参数进行处理。在下一步中，将向 square 方法的方法体中添加一些语句。方法头、花括号以及方法体中的语句一同构成了方法的定义。



软件设计提示
使用方法可增强应用程序的清晰性与结构性。这不仅有助于他人了解开发人员本身所开发的应用程序，而且还有助于开发人员自己完成应用程序的开发、测试以及调试。



好的编程习惯
选择有意义的方法名和参数名可提高应用程序的可读性。

**软件设计提示**

为增强应用程序的可复用性，为每个方法分配一个单一的、明确的任务。

10. 为方法体添加代码 该方法需要对 side 参数值执行一个平方运算，然后，将结果返回给方法的调用程序。将图 12.6 中的第 129 行添加到 square 方法的方法体中。

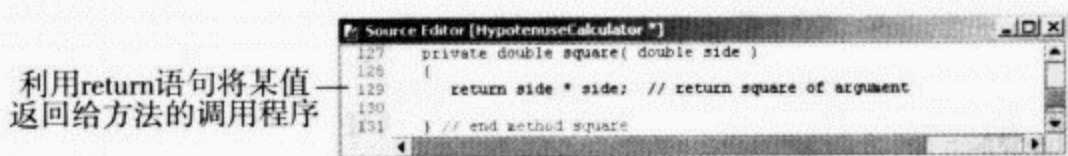


图 12.6 编写 square 方法

第 129 行使用 * 运算符计算 side 的平方值，side 是该方法的参数。第 129 行同时还使用了一条用于返回计算结果的 return 语句。该语句以 return 关键字作为起始，随后是一个表达式。return 语句将返回紧跟在 return 关键字之后的那个表达式的结果——对于本例来说，即返回 side*side 的值，之后，该方法将终止执行。而此值，将被返回到调用该方法的位置处，从而由该方法的调用程序使用。我们将进一步编写有关 square 方法调用的代码。

**好的编程习惯**

方法名应使用动词来命名并以一个小写字母作为起始。除第一个字母外，随后单词中的第一个字母也需大写。

**好的编程习惯**

在多个方法定义之间放置一个空白行可以改善应用程序的可读性。

11. 保存应用程序 保存修改后的源代码文件。

12. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\HypotenuseCalculator 进入到当前的工作目录中。

13. 编译应用程序 通过键入 javac HypotenuseCalculator.java 编译该应用程序。

14. 运行应用程序 若此应用程序能正确编译，通过键入 java HypotenuseCalculator 来运行它。图 12.7 中显示了更新后的应用程序的运行结果。输入边 A 和边 B 的长度，然后点击 Calculate JButton。可以看到，斜边长度并没有被显示出来。我们将在下一个示例中添加代码以实现计算斜边及显示的功能。

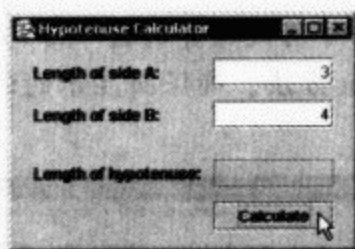


图 12.7 更新后的斜边计算器应用程序

15. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。

16. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

下面，需要向应用程序中添加相应的代码来调用这个新创建的方法。通过多次调用该方法，将会看到代码复用所带来的益处——避免代码的重复性编写。

调用方法

1. 调用 square 方法 下面，将在 calculateJButtonActionPerformed 方法中调用这个刚创建的方法。将图 12.8 中第 123 行至第 129 行添加到应用程序中，这里添加的是一个 if 语句的 else 语句块。第 126 行至第 127 行通过在方法名后跟一对包含该方法参数的圆括号完成对 square 方法的调用。其中，参数为变量 sideA 和 sideB。

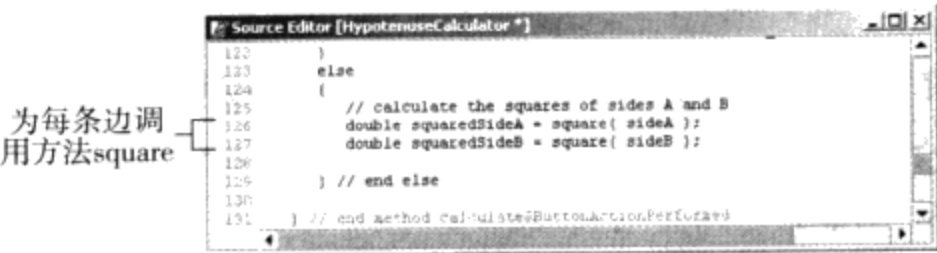


图 12.8 调用 square 方法

所有的方法都需要通过方法来调用(即执行其特定的任务)。方法调用需指明所调用方法的名称,并提供被调用者(被调用的方法)用以接收其完成任务的信息(参数)。当被调用者完成了任务以后,它将把控制权返回给调用者(调用方法)。举例说明,下面是我们曾经调用过的一个典型方法:

```
result = Integer.parseInt( inputJTextField.getText() );
```

其中, Integer.parseInt 为方法名,而 inputJTextField 的 text 属性为传递给该方法的参数。该方法的作用是,利用这个传递的值完成其所定义的任务(将 inputJTextField 的 text 属性作为一个 int 值返回)(注意:当方法拥有参数时,必须为方法调用提供相应的参数值)。

当程序控制到达图 12.8 中的第 126 行时,应用程序便开始调用 square 方法。这时,应用程序将生成 sideA 变量值的一个拷贝(对于本例,假定 sideA 的值为 3),程序控制也将随即转移至 square 方法,并执行 square 方法中的语句。

由 square 方法调用所指定的参数 sideA,表示需要将 sideA 的值(边 A 的长度)的一个拷贝传递给 square; square 将接收这个值的拷贝(3)并将其存储到参数 side 中。当执行到达 square 中的 return 语句时,位于关键字 return 右侧并经过计算所得到的值(3 乘 3 得 9)将返回至第 126 行(如图 12.8 所示),这是调用 square 方法所发生的位置,同时也是这一被调用方法执行结束的位置。程序控制又会转移至该点,随即,应用程序通过把 square 的返回值(9)赋值给变量 squaredSideA,然后继续向下执行。接下来,程序控制将到达第 127 行,并对 square 执行第二次调用,同样的操作会再次发生。利用这一调用,传递给 square 的值将作为变量 sideB 的值(对于本例,假定 sideB 的值为 4),所返回的值(sideB 的平方,即 16)会被赋值给变量 squaredSideB。

- 2. 调用 Java 类库中一个现有的方法 将图 12.9 中第 129 行至第 143 行添加到该应用程序中 if...else 语句的 else 子句里。第 131 行,通过将边 A 的平方与边 B 的平方相加,从而得到斜边的平方,随后将该值赋予变量 squaredHypotenuse。第 135 行,调用的是 Java 类库中 Math 类的 sqrt 方法(通过类名及点运算符确定该方法的调用)。此方法可用做计算斜边平方的平方根从而得到该斜边的长度。第 140 行使用 DecimalFormat 类中内置的 format 方法,确保所显示的斜边只含有两位小数。经格式化处理后的输出随即将显示在第 143 行。

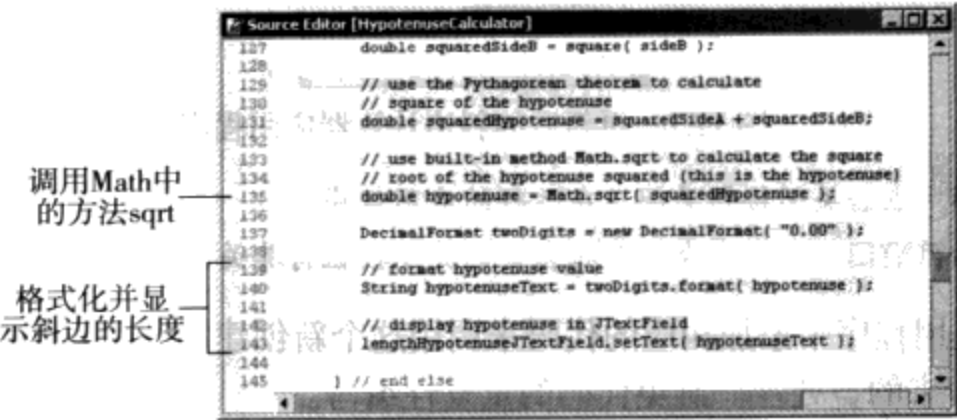


图 12.9 完成方法 calculateJButtonActionPerformed

- 3. 保存应用程序 保存修改后的源代码文件。
- 4. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\HypotenuseCalculator 进入到当前的工作目录中。
- 5. 运行应用程序 若此应用程序能正确编译,通过键入 java HypotenuseCalculator 来运行它。图 12.10 中显示了更新后的应用程序的运行结果。输入边 A 和边 B 的长度,然后按下 Calculate JButton。可以

看到, 显示了斜边的长度。注意, 对于边 A 和边 B 分别为 3 和 4 的直角三角形, 其斜边长度应为 5。还可通过输入直角三角形中边 A 和边 B 的长度 5, 12 以及 8, 15, 对这一应用程序进行测试 (所得到的斜边长度将分别为 13 和 17)。

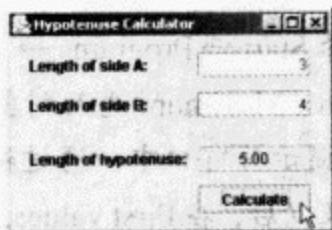


图 12.10 运行完成后的斜边计算器应用程序



错误预防提示

规模较小的方法要比规模较大的方法更易于测试、调试和理解。

6. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。

7. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

到现在为止, 读者已成功地创建出了第一个由程序员定义的方法。通过对这个方法的测试 (运行应用程序), 证实它的确能正常工作。该方法现在可以在任何需要计算某 `double` 数平方的 Java 应用程序中进行使用了。惟一需要做的就是将此方法的定义包含在所需要的应用程序之中。这便是代码复用的一个例子, 利用代码的复用, 程序员便能够更快捷地创建出所开发的应用程序。

正如在斜边计算器应用程序中所演示的那样, 调用某方法的句法应按照下列格式进行:

方法名 (参数列表)

其中, 参数列表是传递至该方法并由逗号分隔的参数所组成的一个列表。参数的数量、类型以及次序都必须符合方法参数列表中所指定参数的要求。如果一个方法的参数列表为空 (即方法名后只有一对空的圆括号), 则表明它不需要任何的参数。

正如在前面例子中所看到的那样, 语句:

`return` 表达式;

实际上可出现在方法语句体中的任何位置, 并把表达式的值返回给该方法的调用程序。方法只能够返回一个值。当执行一条 `return` 语句时, 控制将立即返回至该方法被调用时所发生的位置处, 那里, 便可对所返回的值进行处理 (注意: 表达式可以是任意的 Java 表达式, 甚至还可以是针对另一个方法的调用)。

自测题

1. 方法是通过 _____ 来调用的。

a) 被调用者

b) 调用者

c) 自变量

d) 参数

2. 利用方法中的 _____ 语句可为调用方法返回一个值。

a) `return`

b) `back`

c) `end`

d) 以上答案都不对

答案: 1) b 2) a

12.4 最大值应用程序

下面, 将创建另外一个方法。此方法是最大值应用程序的一部分, 用做返回用户所输入三个数中的最大值。

创建一个返回三个数中最大值的方法

1. 将模板复制到工作目录中 将 C:\Examples\Tutorial12\TemplateApplication\Maximum 目录复制到 C:\SimplyJava 目录中。
2. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\Maximum` 进入到当前的工作目录中。
3. 运行最大值模板应用程序 在命令提示符窗口中，通过键入 `java Maximum` 运行该应用程序（参见图 12.11）。当这个应用程序运行起来以后，在 First value:，Second value: 以及 Third value: JTextField 中输入三个数，然后点击 Maximum JButton。应用程序会确定出三个数中的最大值，并将结果显示在 Maximum: JTextField 中。

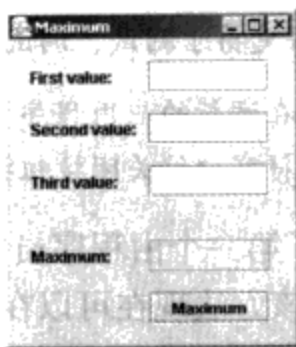


图 12.11 运行最大值应用程序

4. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
5. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。
6. 打开最大值应用程序的模板文件 在自己的文本编辑器中打开模板文件 Maximum.java。
7. 编写 maximumJButtonActionPerformed 方法 该方法是在用户点击 Maximum JButton 时开始执行的，它位于程序第 122 行至第 137 行。将图 12.12 中第 124 行至第 135 行添加到此方法中。第 133 行将调用 determineMaximum 方法，并为其传递用户在应用程序中的 JTextField 内所输入的三个值。此时，方法 determineMaximum 仍未得到定义，而假如现在试图编译该代码的话，则会导致一个语法错误。另外，在方法调用的过程中，如果方法名拼写错误，则同样也会出现一个语法错误。我们将在下一步定义 determineMaximum 方法。

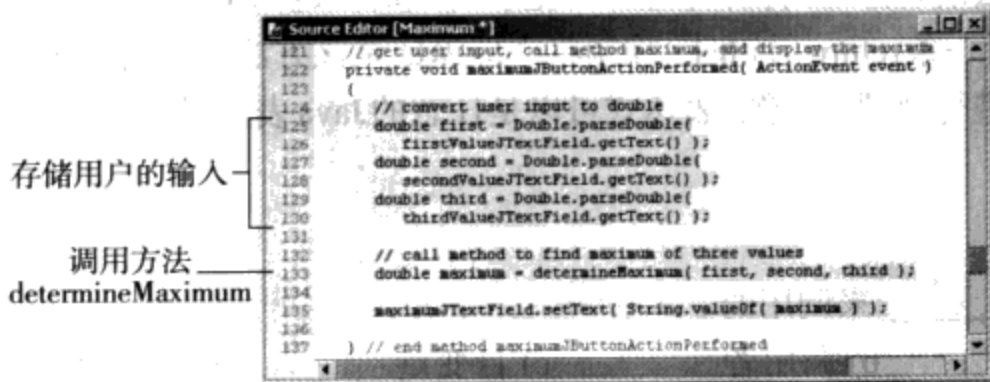


图 12.12 调用 determineMaximum 方法

8. 创建 determineMaximum 方法 将图 12.13 中第 139 行至第 144 行添加到 maximumJButtonActionPerformed 方法的后面。第 140 行是将 determineMaximum 方法的返回类型声明为 double 型的数据。参数列表（参见第 141 行）指明了三个传递给 determineMaximum 方法的参数将分别被存储在参数 one, two 和 three 之中，而所有的这些变量均属于 double 型变量。



常见编程错误

在方法调用过程中，若调用一个并不存在的方法或者是方法名拼写错误，都将导致一个语法错误的出现。

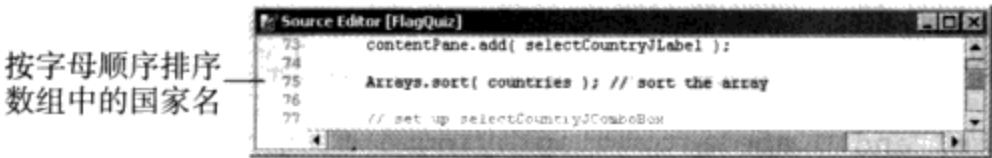


图 16.36 排序国家名数组

注意 sort 方法并无返回值——传递至 sort 方法的数组 countries 将按照“适当位置”进行排序。在此之前，我们为方法所传递的参数，其实只是该参数值的一个拷贝，这种传递方式被称为值传递。采用值传递的方法，被复制的参数值在方法内所做的任何改动都不会影响到原变量中的值。

与之相对应的是第 75 行中所使用了引用传递。当以引用方式传递参数时，实际并未对参数进行复制，而是直接调用了该参数，所以，参数中的数据很可能被修改了。在 Java 中，对象总是按照引用来传递的，而基本类型的参数则是按照值来进行传递的。由于数组也是对象，因此，通过将数组名传递给一个方法可以使该数组按照引用进行传递，从而在内存中访问到原数组中的元素。

- 2. 保存应用程序 保存修改后的源代码文件。
- 3. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\FlagQuiz 进入当前工作目录中。
- 4. 编译应用程序 通过键入 javac FlagQuiz.java 编译该应用程序。
- 5. 运行应用程序 若能正确编译应用程序，通过键入 java FlagQuiz 来运行它。图 16.37 显示了更新后的应用程序的运行结果。此时，应用程序与测试中所使用的完整应用程序具有相同的功能。注意，数组确实是被 Array.sort 方法修改了，因此 selectCountryJComboBox 中的元素现在以字母的顺序来排列了。

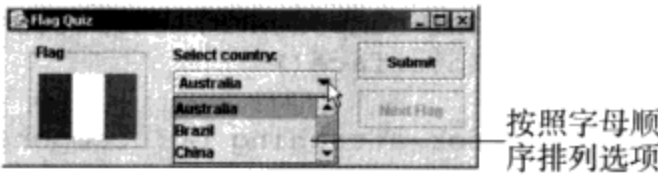


图 16.37 运行国旗知识测评应用程序

- 6. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- 7. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

图 16.38 中给出了国旗知识测评应用程序的完整源代码。本教程中，凡需要添加、查看或是修改的代码，均在图中相应的代码行中进行了突出显示。

```
1 // Tutorial 16: FlagQuiz.java
2 // Quiz the user on their knowledge of flags. The user must try to
3 // match five flags to their countries.
4 import java.awt.*;
5 import java.awt.event.*;
6 import java.util.*;
7 import javax.swing.*;
8 import javax.swing.border.*;
9
10 public class FlagQuiz extends JFrame
11 {
12     // array of country names
13     private String[] countries = { "Russia" , "China" , "United States" ,
14     "Italy" , "Australia" , "South Africa" , "Brazil" , "Spain" };
15
16     // boolean array tracks displayed flags
17     private boolean [] flagsUsed = new boolean [ countries.length ];
18
19     private int currentIndex; // contains the index of current flag
20
```

创建存储国家名的 String 型数组

创建 boolean 型数组

变量 currentIndex 存储当前正在显示的国旗索引

```

21 // tracks the number of flags that have been displayed
22 private int count = 1 ;
23
24 // JPanel and JLabel for displaying a flag image
25 private JPanel flagJPanel;
26 private JLabel flagIconJLabel;
27
28 // JLabel and JComboBox for choosing a country
29 private JLabel selectCountryJLabel;
30 private JComboBox selectCountryJComboBox;
31
32 // JTextField for giving the user feedback
33 private JTextField feedbackJTextField;
34
35 // JButton to submit an answer
36 private JButton submitJButton;
37
38 // JButton to display the next flag
39 private JButton nextFlagJButton;
40
41 // no-argument constructor
42 public FlagQuiz()
43 {
44     createUserInterface();
45 }
46
47 // create and position GUI components; register event handlers
48 private void createUserInterface()
49 {
50     // get content pane for attaching GUI components
51     Container contentPane = getContentPane();
52
53     // enable explicit positioning of GUI components
54     contentPane.setLayout( null );
55
56     // set up flagJPanel
57     flagJPanel = new JPanel();
58     flagJPanel.setBounds( 16 , 8 , 100, 90 );
59     flagJPanel.setLayout( null );
60     flagJPanel.setBorder( new TitledBorder( "Flag" ) );
61     contentPane.add( flagJPanel );
62
63     // set up flagIconJLabel
64     flagIconJLabel = new JLabel();
65     flagIconJLabel.setBounds( 10 , 14, 80 , 80 );
66     flagIconJLabel.setHorizontalAlignment( JLabel.CENTER );
67     flagJPanel.add( flagIconJLabel );
68
69     // set up selectCountryJLabel
70     selectCountryJLabel = new JLabel();
71     selectCountryJLabel.setBounds( 136, 8 , 88, 21 );
72     selectCountryJLabel.setText( "Select country:" );
73     contentPane.add( selectCountryJLabel );
74
75     Arrays.sort( countries ); // sort the array
76
77     // set up selectCountryJComboBox
78     selectCountryJComboBox = new JComboBox( countries );
79     selectCountryJComboBox.setBounds( 136, 32, 135 , 21 );
80     selectCountryJComboBox.setMaximumRowCount( 3 );
81     contentPane.add( selectCountryJComboBox );

```

变量 count 存储
显示国旗的名称

按照字母顺序排序数组中的国家名

自定义

selectCountry-
JComboBox

```
82
83 displayFlag(); // display first flag 显示初始时刻的国旗
84
85 // set up feedbackJTextField
86 feedbackJTextField = new JTextField();
87 feedbackJTextField.setBounds( 136, 64 , 135 , 32 );
88 feedbackJTextField.setHorizontalAlignment(
89     JTextField.CENTER );
90 feedbackJTextField.setEditable( false );
91 contentPane.add( feedbackJTextField );
92
93 // set up submitJButton
94 submitJButton = new JButton();
95 submitJButton.setBounds( 287 , 8 , 88, 32 );
96 submitJButton.setText( "Submit" );
97 contentPane.add( submitJButton );
98 submitJButton.addActionListener(
99
100     new ActionListener() // anonymous inner class
101     {
102         // event handler called when submitJButton is pressed
103         public void actionPerformed((ActionEvent event)
104         {
105             submitJButtonActionPerformed( event );
106         }
107     }
108     ) // end anonymous inner class
109
110 ); // end call to addActionListener
111
112 // set up nextFlagJButton
113 nextFlagJButton = new JButton();
114 nextFlagJButton.setBounds( 287 , 48, 88, 32 );
115 nextFlagJButton.setText( "Next Flag" );
116 nextFlagJButton.setEnabled( false );
117 contentPane.add( nextFlagJButton );
118 nextFlagJButton.addActionListener(
119
120     new ActionListener() // anonymous inner class
121     {
122         // event handler called when nextFlagJButton is pressed
123         public void actionPerformed((ActionEvent event)
124         {
125             nextFlagJButtonActionPerformed( event );
126         }
127     }
128     ) // end anonymous inner class
129
130 ); // end call to addActionListener
131
132 // set properties of application's window
133 setTitle( "Flag Quiz" ); // set title bar string
134 setSize( 390 , 135 ); // set window size
135 setVisible( true ); // display window
136
137 } // end method createUserInterface
138
139 // return an unused random number
140 private int getUniqueRandomNumber()
141 {
142     Random generator = new Random(); 用来创建随机数的对象
```

```

143     int randomNumber;
144
145     // generate random numbers until unused flag is found
146     do
147     {
148         // generate a number between 0-7
149         randomNumber = generator.nextInt( 8 );           确定国旗是否曾显示过
150     }
151     while ( flagsUsed[ randomNumber ] == true );
152
153     // indicate that flag has been used
154     flagsUsed[ randomNumber ] = true ;                   标示未曾显示的国旗并返回其索引
155
156     return randomNumber;
157
158 } // end method getUniqueRandomNumber
159
160 // choose a flag and display it in the JLabel
161 private void displayFlag()
162 {
163     currentIndex = getUniqueRandomNumber(); // get an unused flag      获取未使用国旗的索引
164
165     // create the path for that flag
166     String country =                                     返回与该国旗相对应的国家名
167         ( String ) selectCountryJComboBox.getItemAt( currentIndex );
168     String countryPath = "images/" + country + ".png";      国旗图片的路径名
169
170     // set the flagIconJLabel to display the flag
171     flagIconJLabel.setIcon( new ImageIcon( countryPath ) ); 显示未曾使用过的国旗
172
173 } // end method displayFlag
174
175 // check the answer and update the quiz
176 private void submitJButtonActionPerformed((ActionEvent event) )
177 {
178     // determine whether the answer was correct
179     if ( selectCountryJComboBox.getSelectedIndex()
180         == currentIndex )
181     {
182         feedbackJTextField.setText( "Correct!" );           取得用户答案并显示反馈信息
183     }
184     else // if an incorrect answer is given
185     {
186         feedbackJTextField.setText( "Sorry, incorrect." );
187     }
188
189     // inform user if quiz is over
190     if ( count == 5 )
191     {
192         feedbackJTextField.setText(
193             feedbackJTextField.getText() + " Done!" );
194         nextFlagJButton.setEnabled( false );
195         submitJButton.setEnabled( false );
196         selectCountryJComboBox.setEnabled( false );
197     }                                           判断测验是否结束
198     else // if less than 5 flags have been displayed
199     {
200         submitJButton.setEnabled( false );
201         nextFlagJButton.setEnabled( true );
202     }

```



```
203
204 } // end method submitJButtonActionPerformed
205
206 // display next flag in the quiz
207 private void nextFlagJButtonActionPerformed((ActionEvent event) )
208 {
209     displayFlag(); // display next flag           显示下一面需用户识别的国旗
210     count++;
211
212     // reset GUI components to initial states
213     feedbackJTextField.setText( "" );
214     selectCountryJComboBox.setSelectedIndex( 0 );  设置JComboBox以显示其第一个选项
215     submitJButton.setEnabled( true );
216     nextFlagJButton.setEnabled( false );
217
218 } // end method nextFlagJButtonActionPerformed
219
220 // main method
221 public static void main( String[] args )
222 {
223     FlagQuiz application = new FlagQuiz();
224     application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
225
226 } // end method main
227
228 } // end class FlagQuiz
```

图 16.38 国旗知识测评应用程序的源代码

自测题

1. 将数组中的元素按照某种顺序来排列的过程称为数组的 ____。
a) 创建 b) 排序 c) 声明 d) 初始化
2. 下面哪一个方法可用于排序数组 `averageRainfall`?
a) `Arrays(averageRainfall).sort()` b) `sort.Arrays(averageRainfall)`
c) `sort(averageRainfall)` d) `Arrays.sort(averageRainfall)`

答案: 1) b 2) d

16.6 小结

在本教程中,学习了一种被称之为数组的数据结构,利用它来存储相同类型的元素。之后,学习了如何创建并初始化一维数组,以及如何通过使用一个索引访问数组中的数据(数组中第一个元素的索引为0)。表达式 `arrayName.length` 可用来取得数组 `arrayName` 中的元素个数。在此基础之上,我们创建了一个简单的数组求和应用程序,计算了存储在数组中所有 `int` 值的总和。随后,研究了国旗知识测评应用程序的伪代码表示以及相应的 ACE 表,实现了对该应用程序的创建。

在创建国旗知识测评应用程序的过程中,向读者介绍了 `JComboBox` 组件。学习了如何将 `JComboBox` 组件添加到应用程序中以及如何修改 `JComboBox` 的外观等内容。随后,利用数组向 `JComboBox` 中装载了所需要的数据。

通过使用 `Arrays.sort` 方法,懂得如何将数组中的元素按照字母顺序进行排序,以及方法中值传递与引用传递之间的区别。同时,也明白了数组也是一种对象,在方法传递过程中采用的是引用传递,即通过方法调用可修改其中的数组参数。

在下一个教程中,将学习如何创建更为复杂的数组,这种数组可看做是由包含行与列的数据所组成的,称它们为二维数组。将通过二维数组创建一个成绩评定应用程序。

技术小结

返回 JComboBox 中索引为 n 的选项。

- 使用 `getIndexOfItemAt` 方法。

创建数组

- 使用下面的格式声明数组:

```
arrayType[] arrayName;
```

其中:

`arrayName` 为数组的引用名, `arrayType` 为数组中存储数据的类型。

- 使用下面的表达式创建数组:

```
arrayName = new arrayType[size];
```

其中:

`size` 表示数组中的元素个数。当数组通过此方法来创建时,数组中的元素将被初始化为所存类型的默认值。

- 通过下面的表达式声明并初始化数组:

```
arrayType[] arrayName = { arrayInitializerList };
```

其中:

`arrayInitializerList` 为一列由逗号隔开的值,利用它们来完成对数组的初始化操作。

查找数组中的第 n 个元素

- 使用索引 n 。
- 将此索引封闭在一对位于数组名之后的方括号内。

取得数组长度

- 使用表达式 `arrayName.length`, 其中 `arrayName` 为所要查找其长度的数组名。

为用户提供多个选项

- 使用 JComboBox 组件。

设置 JComboBox 一次显示的最大选项数

- 使用 `setMaximumRowCount` 方法。

取得 JComboBox 内的用户选择

- 使用 `getSelectedIndex` 方法。

排序数组

- 调用 `Arrays.sort` 方法并为其传递所要排序的数组。

关键术语

数组 包含相同类型元素的一种数据结构。

数组边界 确定数组元素索引的整数值范围。数组下限为 0, 上限为数组长度减 1。

数组初始化器 一系列由逗号分隔且封装在花括号对 (`{ }`) 内的表达式,用以实现数组中元素的初始化操作。当初始化器为空时,则数组中的元素将被初始化为数组类型的默认值。

表达式 `arrayName.length` 能够计算出数组内的元素个数。

Arrays.sort 将参数数组中的元素按照升序或者是字母表的顺序进行排序。

数据结构 用来分组并组织相关数据的一项技术。

元素 数组中的条目。

JComboBox的getItemAt方法 接收一个代表某个索引的int型参数并返回该索引位于JComboBox中的对象。

JComboBox的getSelectedIndex方法 返回被选择选项的索引。

索引 数组元素的位置编号,也称为下标。索引可以是0,正整数或者是能够返回0或正整数的表达式。

如果应用程序使用表达式作为其索引,则表达式将首先得到计算。

带索引的数组名 数组名后有一个封闭在方括号内的索引。带索引的数组名可以放置在赋值语句的左侧,从而实现对数组元素的赋值;也可以将它放置在赋值语句的右侧,以此得到该数组元素的值。

JComboBox组件 在下拉列表中为用户提供选项。

一维数组 只使用一个索引的数组。

引用传递 能够通过方法调用来修改其中的参数。

值传递 能够将参数的一个拷贝传递给某个方法,因而不会对原数据进行访问和修改。

位置编号 代表数组中某个特定位置的值。位置编号起始于0。

JComboBox的setMaximumRowCount方法 指定下拉列表内一次可以显示的选项数。

JComboBox的setSelectedIndex方法 设置JComboBox中被选选项的索引。

排序 按照一定顺序(比如,升序或降序)对数据进行的重新组织。

下标 索引的另一种称呼。

第0号元素 数组中的首个元素。

GUI 设计导航

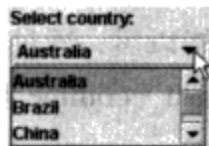
JComboBox

- 每一个JComboBox组件都应具有一个描述其内容的JLabel。
- 对拥有多个选项的JComboBox进行排序将有助于用户查找到所需要的选项。

Java 类库索引

JComboBox 该组件允许用户从下拉列表中选择需要的选项。

- 运行



- 构造方法

JComboBox 接收一个数组作为参数。数组中的元素将被装载到JComboBox中。

```
private String[] countries = { "Russia", "China", "United States",  
    "Italy", "Australia", "South Africa", "Brazil", "Spain" };  
selectCountryJComboBox = new JComboBox( countries );
```

- 方法

setMaximumRowCount 设置JComboBox下拉列表内一次可显示的条目数。

setBounds 指定JComboBox组件相对于容器组件左上角的位置和大小。

getItemAt 接收一个代表索引的int型参数并返回该索引位于JComboBox中的值。

getSelectedIndex 返回JComboBox内被选择选项的索引。

setSelectedIndex 设置JComboBox内被选择选项的索引。

Arrays 此类提供许多管理数组的方法。

- 方法

sort 完成对数组元素的排序。数值型数组将按升序进行排序,字符型数组则按照字母顺序进行排序。

习题

选择题

- 16.1 数组可以被声明为 _____ 类型。
a) double b) int c) String d) 任意
- 16.2 数组中的元素之所以是相互有联系的是因为它们具有相同的名称和 _____。
a) 常量值 b) 下标 c) 类型 d) 值
- 16.3 表达式 _____ 可返回数组的最大索引。
a) arrayName.getUpperBound b) arrayName.getUpperLimit
c) arrayName.length d) arrayName.length - 1
- 16.4 数组中的第一个元素称 _____。
a) 下标 b) 第 0 个元素 c) 数组长度 d) 数组中的最小值
- 16.5 数组 _____。
a) 是组件 b) 总是一维的 c) 任何时候都处于已排序的状态 d) 是对象
- 16.6 使用 _____ 符号可创建作为指定数组中元素初始值的初始化器。
a) [和] b) < 和 > c) (和) d) { 和 }
- 16.7 下面哪一个方法能够将数组 words 中的元素按照字母顺序进行排序？
a) Arrays.sort(words) b) words.sortArray()
c) Arrays.sort(words,1) d) sort(words)
- 16.8 利用 _____ 方法可以设置 JComboBox 下拉列表中一次可以显示的选项数。
a) getUpperBound b) getItemAt c) setMaximumRowCount d) setBounds
- 16.9 当向方法内传递一个参数时，将该参数值的一个拷贝传递给方法的传递方式称为 _____。
a) 调用传递 b) 值传递 c) 引用传递 d) 方法传递
- 16.10 _____ 方法用于返回 JComboBox 中所选择选项的索引。
a) getUpperBound b) getSelectedIndex c) setMaximumRowCount d) getItemAt

练习题

- 16.11(改进的国旗知识测评应用程序) 对本教程中的国旗知识测评应用程序进行改进，统计用户回答问题正确的数目并在所有问题回答完毕以后，通过 JTextField 内显示一条信息，描述用户回答问题的情况（参见图 16.39）。该 JTextField（commentJTextField）已在模板中得到添加。下表中列出了可以显示的信息（参见图 16.40）。
- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial16\Exercises\EnhancedFlagQuiz 目录复制到 C:\SimplyJava 目录中。

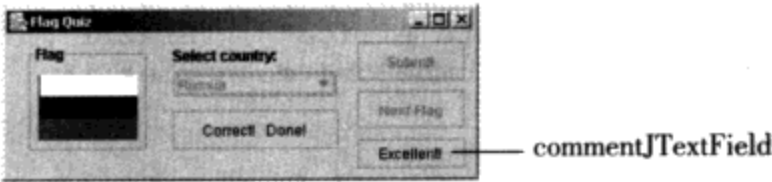


图 16.39 改进的国旗知识测评应用程序的 GUI

问题回答正确的数目	显示信息
5	Excellent!
4	Very good!
3	Good.
2	Poor.
1 或 0	Fail

图 16.40 显示给用户的信息

- b) 打开模板文件 在自己的文本编辑器中打开 FlagQuiz.java 文件。
 - c) 添加统计正确答案的变量 在第 24 行, 加入一行注释, 说明即将创建的变量用于存储用户回答问题正确的数目。在第 25 行, 声明一个 int 型变量 correct 并将其初始化为 0。
 - d) 统计正确答案 在第 196 行, 自增 correct 变量。该语句会在每次提交一个正确答案后自增变量 correct。
 - e) 显示消息 在第 209 行至第 227 行添加一条 switch 语句, 根据 correct 变量的值在 commentJTextField 中显示一条适当信息。
 - f) 保存应用程序 保存修改后的源代码文件。
 - g) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\EnhancedFlagQuiz 进入到当前工作目录中。
 - h) 编译应用程序 通过键入 javac FlagQuiz.java 编译该应用程序。
 - i) 运行应用程序 若能正确编译应用程序, 键入 java FlagQuiz 来运行它。多次运行该应用程序, 每次输入数目不同的正确答案, 确保应用程序结束时 commentJTextField 中可以显示出正确的信息。
 - j) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
 - k) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。
- 16.12 (薪金调查应用程序) 通过一维数组解决下列问题: 某公司按照佣金额多少为销售人员发放工资。销售人员的工资额为每周 \$200 加本周内销售总额的 9%。例如, 某销售人员一周内的销售总额为 \$5000, 那么他的工资则为 $\$200 + \$5000 * 9\%$, 得 \$650。编写这样的一个应用程序 (使用计数器数组), 确定出不同工资范围内销售人员的总数 (假设每位销售人员的工资均被截取为整数值), 工资额的范围包括: \$200~\$299, \$300~\$399, \$400~\$499, \$500~\$599, \$600~\$699, \$700~\$799, \$800~\$899, \$900~\$999 以及大于 \$999。
- 该应用程序允许用户通过 JTextField 输入每位员工的销售总额, 并通过点击 Calculate JButton 计算出该名销售人员的工资额。当用户输入所有信息以后, 点击 Show Totals JButton 便可显示出上述每个范围内销售人员的总数目。完成后的应用程序请参照图 16.41。在该图中, 分别输入销售额 \$1230 和 \$5406, 得到的工资分别为 \$310.70 和 \$686.54, 而相应的工资额范围 (\$300~\$399 和 \$600~\$699) 则在点击 Show Totals JButton 之后加 1。

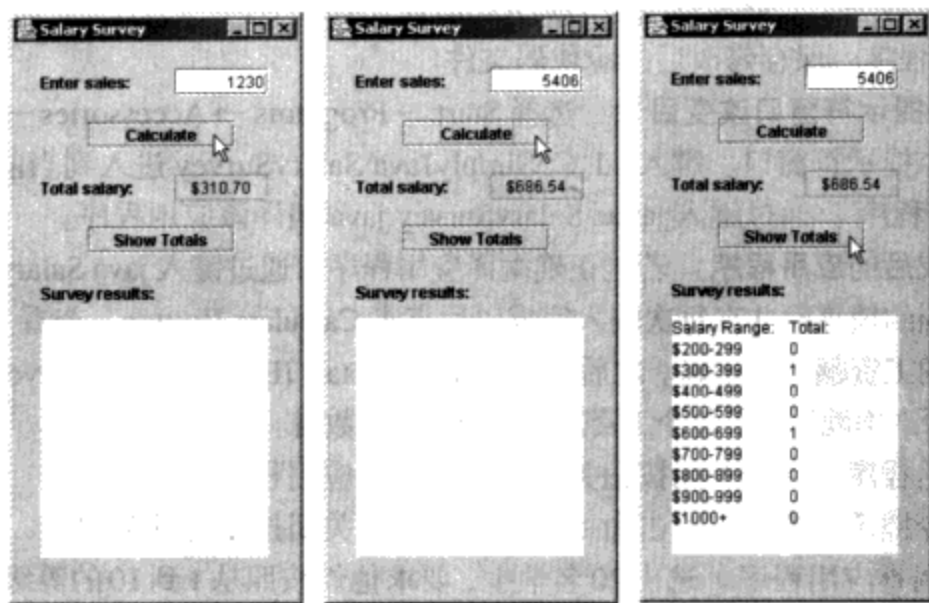


图 16.41 薪金调查程序应用程序的 GUI

- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial16\Exercises\SalarySurvey 目录复制到 C:\SimplyJava 目录中。
- b) 打开模板文件 在自己的文本编辑器中打开 SalarySurvey.java 文件。
- c) 创建代表不同工资额范围内员工人数的数组 在第 32 行至第 33 行创建一个 int 型数组 resultArray, 存储不同薪金范围内的员工总人数。第 32 行为注释行, 第 33 行为具体创建该数

组的代码行。resultArray 中的元素代表不同的工资额范围。在使用关键字 new 的同时, 将数组 resultArray 的长度定为 11。该应用程序一共有 9 个工资范围, 但为了方便起见, 这里使用了一个范围较大的数组 (长度大小为 11)。我们需要忽略数组中的前两个元素, 因此可以使用的元素索引值范围为 2~10, 每个索引位置代表了一个工资额范围 (如位置 2 代表 \$200~\$299, 位置 3 代表 \$300~\$399, 依次类推)。数组的正确索引可以通过将工资额除以 100 (利用整数间的除法运算) 来得到。例如, 工资额 \$350 除以 100 得到结果 (3) 将作为访问该数组中代表 \$300~\$399 范围的索引。但是, 这里仍存在一个例外, 当工资额超过 \$1000 时, 除以 100 的结果将大于或等于 10。此时, 需要在应用程序中单独处理该范围内的工资额。或许我们已经注意到, 忽略数组 resultArray 中的前两个元素会使编程变得更为方便和快捷。如果使用索引 0~8, 那么, 每次都需要减 2 (除以 100 之后) 才可计算出正确的索引值!

- d) **排序应用程序的运行结果** 第 136 行将根据输入的销售额计算员工的工资额并将结果存储在变量 salary 中 (已在模板中提供)。我们需要对代表正确工资额范围的 resultArray 中的元素值进行自增。刚才提到过, 为了访问 resultArray 中的正确位置, 需要将员工的工资额除以 100。在第 137 行, 声明一个 int 型变量 index, 存储工资额除以 100 以后的值。需要将此结果转换为 int 类型。在第 139 行至第 146 行通过定义一条 if...else 语句, 对 resultArray 中的适当元素进行自增。利用 index 中的值作为自增运算时的 resultArray 中的位置索引。切记, 属于最后范围内的工资额, 其 index 值将超过 10。
- e) **显示工资额范围** 下面, 添加用于显示 resultArray 中值的代码。在第 160 行, 添加一个能够从 2 迭代至 9 的 for 语句首部。这条 for 语句将用于显示前 8 个范围内的结果, 其中的计数器 (i) 将作为 resultArray 的索引值。最后一个工资额范围将作为一个特殊情况在该 for 语句之后单独进行处理。在第 161 行, 添加一个左花括号作为该 for 语句体的起始。在第 162 行至第 163 行初始化变量 lowerBound 与 upperBound (声明于第 155 行至第 156 行)。将变量 lowerBound 和变量 upperBound 分别设置为当前薪金范围的下限和上限。例如, 如果 i 的当前值为 4, 那么下限值为 400 而上限值为 499。在第 165 行至第 166 行, 使用 append 方法将相应的文本追加到 resultJTextArea 中。通过 lowerBound, upperBound 和 resultArray 中的值在 JTextArea 中实现不同工资额范围的显示。参照图 16.41, 在输出结果中添加相应的美元符、连字符、tab 符以及换行符。在第 167 行, 通过添加一个右花括号结束该 for 语句的语句体。在第 169 行, 利用 append 方法将最后一个工资额范围添加到 resultJTextArea 中。
- f) **保存应用程序** 保存修改后的源代码文件。
- g) **打开命令提示符窗口改变目录** 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\SalarySurvey 进入到当前工作目录中。
- h) **编译应用程序** 通过键入 javac SalarySurvey.java 编译该应用程序。
- i) **运行完成后的应用程序** 若能正确编译应用程序, 通过键入 java SalarySurvey 来运行它。输入一些不同的销售额并在每次输入完成以后点击 Calculate JButton, 查看 Total salary:JTextField 中所显示的工资额是否正确。之后, 点击 Show Totals JButton, 查看 Survey results:JTextArea 中的输出是否准确地反映了每个工资范围内的人员总数目。
- j) **关闭应用程序** 点击关闭按钮关闭正在运行的应用程序。
- k) **关闭命令提示符窗口** 点击命令提示符窗口的关闭按钮将其关闭。

16.13 (食堂问卷调查应用程序) 选出 20 名学生, 要求他们按照从 1 到 10 的等级, 对食堂内的饭菜质量进行评定, 其中, 等级 1 代表 "awful" (糟透了), 等级 10 代表 "excellent" (好极了)。用户将通过 JComboBox 输入不同等级。该应用程序利用一个 int 型数组存放 20 个学生的评定结果并判断出每一个等级所出现的频率。最后, 将得到的评定结果以柱状图的形式显示在一个多行 JTextField 中。柱状图 (又称条形图) 是一张通过条形图来显示相应数值的图表。图中, 较长的条形图代表较大的数值。使用图形化方式来表示柱状图中的数值, 有一种较为简单方式, 即通过星号 (*) 表示其中的数值大小。图 16.42 中显示了完成后的应用程序的运行结果。

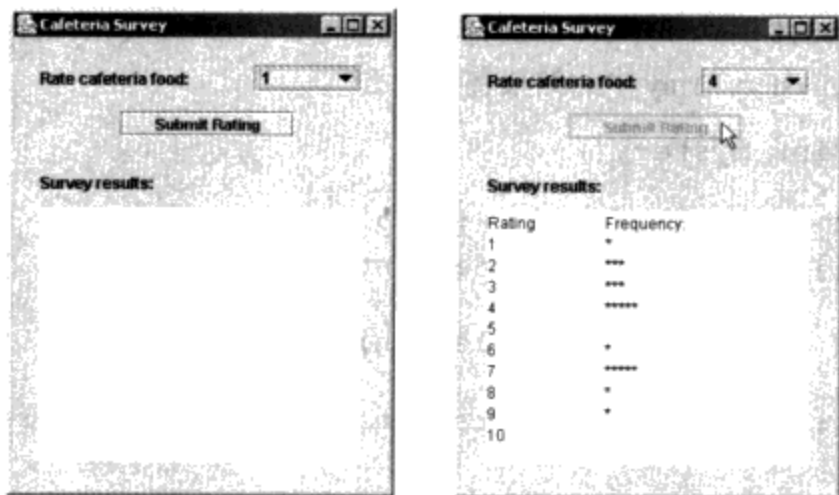


图 16.42 食堂问卷调查应用程序的 GUI

- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial16\Exercises\CafeteriaSurvey 目录复制到 C:\SimplyJava 目录中。
- b) 打开模板文件 在自己的文本编辑器中打开 CafeteriaSurvey.java 文件。
- c) 创建存储所有可能等级的数组 在第 23 行至第 25 行, 创建一个存储多个选项的 String 型数组, 该数组中含有 10 个字符串形式的连续整数 (如 "1", "2", 等等), 其范围为 1~10 (包括 10)。第 23 行作为注释行, 第 24 行至第 25 行用于创建并初始化该数组。
- d) 创建一个存储相应结果的数组 在第 27 行至第 28 行, 创建一个长度为 11 的 int 型数组 responses。此数组将用来存储 10 个等级中每一个等级所对应的数值大小 (0 元素未被使用)。第 27 行为注释行, 第 28 行用于创建此数组。
- e) 自定义 ratingJComboBox 在第 54 行通过自定义 ratingJComboBox 显示所有可能的等级。
- f) 排序数组 responses 现在, 查看一下方法 submitRatingJButtonActionPerformed, 此方法在用户点击 Submit Rating JButton 时执行。第 101 行通过自增变量 responseCounter (已在模板中提供), 存储用户输入的评定总数。在随后的第 102 行, 将输入的评定等级存储到了变量 input 中。之所以先将 ratingJComboBox.getSelectedIndex 的结果加 1 是因为该 JComboBox 的下标是从 0 开始的。此时, 变量 input 中的值为数组 responses 中某一等级的索引。通过第 103 行的变量 input 对数组 responses 中的适当元素加 1。
- g) 显示柱状图 下面, 需以柱状图的形式显示出评定结果。第 106 行开始的一条 if 语句会在输入完 20 个结果以后开始执行。在这条 if 语句体内, 定义了一条 for 循环语句 (位于第 110 行至第 116 行)。该语句用来循环每一个评分的等级并通过第 112 显示该等级 (后跟一个 tab 符) 以及通过第 114 行加入一个换行符。我们将在每一个等级的右边添加所要显示的星号。将鼠标指针放置在第 114 行, 添加一个从 1 循环至当前等级的选票数 (在数组 responses 中存储) 的 for 循环。在第 115 行, 添加一个起始该 for 语句体的左花括号。在第 116 行, 将一个星号添加到输出中。由于该 for 语句所执行的循环次数与当前等级的投票数相同, 因而可以显示出正确的星号数。在第 118 行, 通过添加一个右花括号来结束该 for 循环。在此循环右花括号的后面, 再添加一行表示该 for 循环已经结束的注释。
- h) 保存应用程序 保存修改后的源代码文件。
- i) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\CafeteriaSurvey 进入当前的工组目录中。
- j) 编译应用程序 通过键入 javac CafeteriaSurvey.java 编译该应用程序。
- k) 运行完成后的应用程序 若能正确编译应用程序, 通过键入 java CafeteriaSurvey 来运行它。从 ratingJComboBox 中分别选出 20 个不同等级并依次点击 Submit Rating JButton。查看 resultJTextArea 中的每一个等级是否包含了正确数目的星号。
- l) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- m) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

16.14 (说出这段代码的作用) `mystery` 方法将利用 `numbers` 数组修改 `mysteryArray` 数组中的元素。到方法结束以后, `mysteryArray` 中的结果将是什么?

```

1 private int mystery()
2 {
3     int [] numbers = { 0 , 1 , 2 , 3 , 4 };
4     int [] mysteryArray = new int[ numbers.length ];
5
6     for ( int i = numbers.length; i > 0 ; i-- )
7     {
8         mysteryArray[ numbers.length - i ] = numbers[ i - 1 ];
9     }
10
11 } // end method mystery

```

16.15 (找出代码中的错误) 下列代码将通过一个 `for` 循环求出数组中的元素总和。请找出代码中的错误。

```

1 public void sumArray()
2 {
3     int [] numbers = new int [] { 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 };
4
5     for ( int counter = 0 ; counter <= numbers.size; counter++ )
6     {
7         int sum += numbers[ counter ];
8     }
9 } // end method sumArray

```

挑战题

16.16 (道路标志测验应用程序) 编写一个应用程序, 测试用户对道路标志知识的了解。该应用程序随机显示一个路标并要求用户从 `JComboBox` 中选出此路标的名称。应用程序的运行结果如图 16.43 中所示 (可以看到, 该应用程序同国旗知识测评应用程序非常类似)。表示道路标志的图片位于 `C:\Examples\Tutorial16\Exercises\RoadSignTest\images` 目录下。

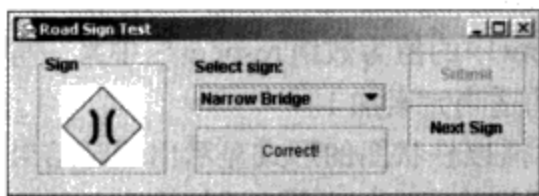


图 16.43 道路标志测验应用程序的 GUI

- 将模板复制到工作目录中 将 `C:\Examples\Tutorial16\Exercises\RoadSignTest` 目录复制到 `C:\SimplyJava` 目录中。
- 打开模板文件 在自己的文本编辑器中打开 `RoadSignTest.java` 文件。
- 声明一个包含用户选项的数组 在第 29 行至第 33 行, 创建一个 `String` 型数组 `signs` (第 29 行为注释行, 第 30 行至第 33 行用来创建并初始化该数组)。数组中将包含以下内容: "Do Not Enter", "Narrow Bridge", "No Bicycles", "No Left Turn", "No Pedestrians", "No U-turn", "Road Narrows", "Stop", "Stop Sign Ahead", "Traffic Signals Ahead", "Winding Road Ahead", "Yield"。
- 声明一个对已使用过的路标进行存储的数组 在第 35 行至第 36 行, 声明一个 `boolean` 型数组 `signsUsed` (第 35 行为注释行, 第 36 行用来创建此数组), 指定该数组的长度为 12。
- 排序 `signs` 数组 在第 73 行, 按照字母顺序排序 `signs` 数组。
- 自定义 `signJComboBox` 修改第 76 行, 通过 `signJComboBox` 显示 `signs` 数组中提供的所有选项。在第 78 行, 将 `signJComboBox` 自定义为每次最多只显示 4 个选项。
- 修改 `getUniqueRandomNumber` 方法 方法 `getUniqueRandomNumber` 通过数组 `signsUsed` 并借助 `Random` 的对象 `generator` 生成一个未曾显示过的随机路标。我们需要添加相应的代码来产生一个未曾使用过的随机数。在第 142 行至第 147 行, 定义一条 `do...while` 语句, 产生一个 0~11

范围内的随机数。该do...while循环语句将一直循环至一个未曾使用过的随机数。在第149行，修改 signsUsed 数组，指定一个由新生成的随机数所对应的路标。

- h) **保存应用程序** 保存修改过源代码应用程序。
- i) **打开命令提示符窗口改变目录** 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\RoadSignTest 进入到当前工作目录中。
- j) **编译应用程序** 通过键入 javac RoadSignTest.java 编译该应用程序。
- k) **运行完成后的应用程序** 若能正确编译应用程序，通过键入 java RoadSignTest 来运行它。为确保显示出正确的反馈信息，分别输入正确和错误的答案并查看 JComboBox 中是否一次最多显示 4 个选项。
- l) **关闭应用程序** 点击关闭按钮关闭正在运行的应用程序。
- m) **关闭命令提示符窗口** 点击命令提示符窗口的关闭按钮将其关闭。

教程 17 成绩评定应用程序

介绍二维数组及 JRadioButton 组件



教学目标

在本教程中，读者将学到以下内容：

- 一维数组与二维数组间的差别
- 声明并操作二维数组
- 二维数组的应用
- 为用户提供惟一选项的 JRadioButton

在本教程中，我们将向读者介绍有关二维数组的概念。二维数组同一维数组类似，它们都能存储多个数值。然而，开发人员可通过二维数组，以行为单位来存储数据。同时，本教程还将向读者介绍有关 JRadioButton 组件的使用，即利用该组件使用户只能选择一个选项。

17.1 探试成绩评定应用程序

在这一教程里，将通过使用二维数组，完成一个成绩评定应用程序。该应用程序必须满足下面的需求：

应用程序需求分析

某教师将对一个有 10 名学生的班级进行三次测验。其中，测验成绩是一些 0~100 之间的整数。该教师希望能开发一个应用程序，用于计算每位学生的平均分以及班级内全体同学总的平均分。该教师同时还希望有一个能够按数字分值或者是字母分值来查看成绩的选项。字母分值将根据下面的成绩系统进行换算：

90~100	A
80~89	B
70~79	C
60~69	D
60 以下	F

应用程序允许用户输入每位学生的姓名以及三次测验的成绩。之后，便可计算出每位学生的平均分和全班总的平均分。在默认情况下，应用程序是按照数字分值来进行显示的。

每位学生的平均分，等于该名学生的三次测验成绩的总和除以 3。班级平均分，等于全体学生的平均分总和除以该班级学生的总人数（本例中班级总人数为 10 人）。我们将以这个完成后的应用程序的探试作为起点。之后，将学习另外一些 Java 技术，并最终创建出一个属于自己的应用程序。



探试成绩评定应用程序

1. 定位到完成后的应用程序 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\Examples\Tutorial17\CompletedApplication\StudentGrades` 将目录改变到这个完成后的成绩评定应用程序的目录下面。
2. 运行成绩评定应用程序 在命令提示符窗口下通过键入 `java StudentGrades` 运行该应用程序 (参见图 17.1)。
3. 输入数据 在 Student Name:JTextField 中输入 Greta Green。在 “Test 1:”, “Test 2:” 和 “Test 3:” JTextField 中分别输入 87, 94 和 93 (参见图 17.2)。点击 Submit Grades JButton, 这时, 数据会显示在 JTextArea 中 (参见图 17.3)。

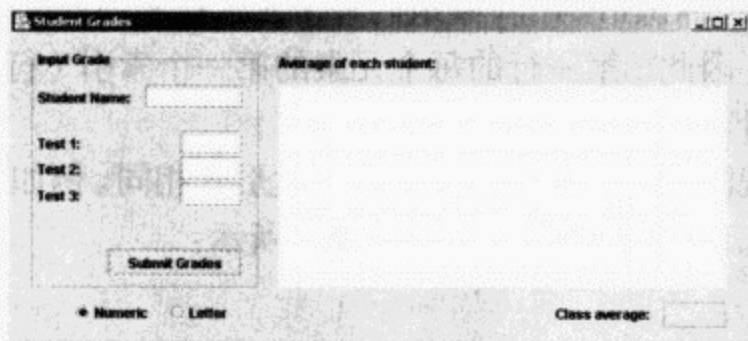


图 17.1 运行完成后的成绩评定应用程序

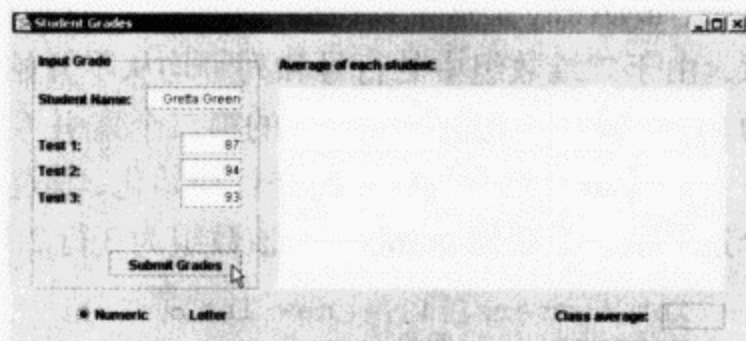
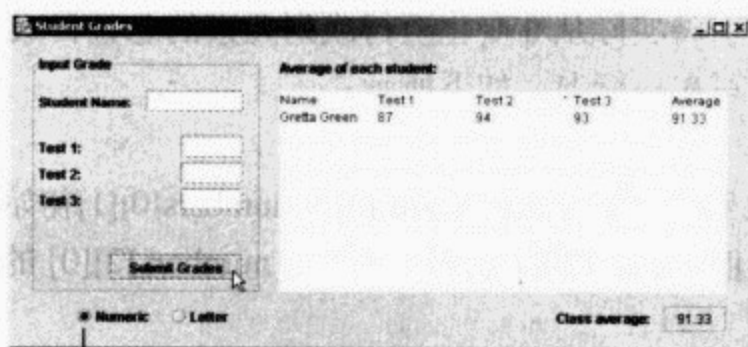


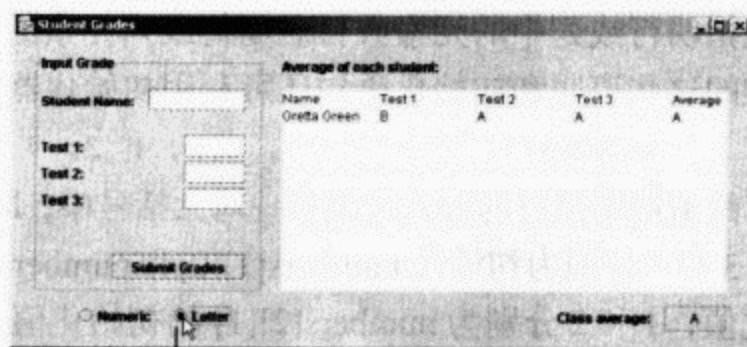
图 17.2 向成绩评定应用程序中输入数据

4. 改变 JTextArea 的显示结果 通过点击 Letter JRadioButton 改变 JTextArea 的显示结果 (参见图 17.4)。此时, JTextArea 中将显示出字母分值系统的数据。点击 Numeric JRadioButton, 可再次将数据显示为数值型格式 (参见图 17.3)。



选定 Numeric
JRadioButton
作为默认选项

图 17.3 显示学生的数值成绩



选取 Letter
JRadioButton

图 17.4 显示学生的字母成绩

5. 关闭正在运行的应用程序 点击关闭按钮关闭正在运行的应用程序。
6. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

17.2 二维数组

在学习二维数组之前, 我们已经学过一维数组 (用来包含一系列值)。从这一节开始, 将学习有关二维数组的概念, 该类型的数组需通过两个索引来标识指定的元素。二维数组通常被用于表示由多个值构成的一张表, 即利用行和列的形式组织信息。每一行的大小相同, 因而具备相同的列数。为标识其中特定的表元素, 必须指定两个下标。习惯上, 用第一个下标标识元素的行, 用第二个下标标识元素的列。图 17.5 中列举了一个二维数组 `myArray`。该数组为一个 3 行 4 列的数组。一个具有 m 行 n 列的二维数组被称为 $m \times n$ 数组; 因此, 图 17.5 中的这个数组就是一个 3×4 数组。

	列0	列1	列2	列3
行0	myArray[0][0]	myArray[0][1]	myArray[0][2]	myArray[0][3]
行1	myArray[1][0]	myArray[1][1]	myArray[1][2]	myArray[1][3]
行2	myArray[2][0]	myArray[2][1]	myArray[2][2]	myArray[2][3]

列索引 (或称下标)
 行索引 (或称下标)
 数组名

图 17.5 一个 3 行 4 列的二维数组

可以看到, 图 17.5 中 myArray 数组内的每个元素都是按照 myArray[i][j] 的形式来进行标识的。其中, myArray 是这个数组的数组名, i 和 j 是惟一标识 myArray 数组内每个元素行和列的索引。注意, 由于二维数组中的行号和列号均从零开始, 因此, 第一行的每个元素的第一个索引 (行索引) 为 0, 而最后一列的每个元素的第二个索引 (列索引) 则为 3。

二维数组在声明的同时进行初始化, 其过程同一维数组中所使用的表达形式相同。例如, 有一个这样的二维数组 numbers, 该数组为 3 行 2 列, 其声明及初始化过程如下所示:

```
int numbers[][] = new int [ 3 ][ 2 ];
numbers[ 0 ][ 0 ] = 1 ;
numbers[ 0 ][ 1 ] = 2 ;
numbers[ 1 ][ 0 ] = 3 ;
numbers[ 1 ][ 1 ] = 4 ;
numbers[ 2 ][ 0 ] = 5 ;
numbers[ 2 ][ 1 ] = 6 ;
```

可以看到, 该数组在创建时, 用于指定行数与列数目的整数 (此例中为 3 和 2), 总是能够确切地指明行或列中的元素数目。而行或列元素的索引将实际从 0 变化至行或列元素的总数目减 1。上述的这些声明和初始化语句还可利用初始化器书写在一行上, 如下所示:

```
int numbers[][] = { { 1, 2 }, { 3, 4 }, { 5, 6 } };
```

所有的值都将以行的形式组织在一对大括号中, 即 1 和 2 分别为 numbers[0][0] 和 numbers[0][1] 初始化值, 3 和 4 分别为初始化 numbers[1][0] 和 numbers[1][1] 初始化值, 5 和 6 分别为 numbers[2][0] 的初始化值, 1 和 2 分别为 numbers[2][1] 的初始化值。

自测题

- 使用两个索引的数组被称为 _____ 数组。
a) 单下标 b) 二维 c) 正方 d) 一维
- 能正确创建一个 2 行 5 列的 int 型数组的语句是 _____。
a) new integer[2][5]; b) new integer[5][2]; c) new int[2][5]; d) new int[1][4];


答案: 1) b 2) c

17.3 学习使用 JRadioButton

组件 JRadioButton 的外观是一个小圆圈, 其内部可为空 (未被选取时) 或者被黑色圆点填充 (被选取时)。JRadioButton 被认为是一种状态按钮, 因为它只能处于“开”状态或者“关”状态。我们曾在教程 7 中学过另外一种状态按钮——JCheckBox。

通常需要将多个 JRadioButton 组织在一个可容纳任意 JRadioButton 数量的 ButtonGroup 中。起初, 每一个 ButtonGroup 中只能有零或一个处于选取状态的 JRadioButton。一旦有 JRadioButton 被选取, 则该 ButtonGroup 将确保一次只能有惟一的一个 JRadioButton 处于被选取状态。当这个

ButtonGroup 中另有一个 JRadioButton 被选取时, 前面那个已选取的 JRadioButton 此时将被撤销。注意, JRadioButton 中并不存在什么默认的 ButtonGroup, 因此, 开发人员必须明确地将每个 JRadioButton 添加到一个指定的 ButtonGroup 中。同样, 也需要为自己应用程序中的 JRadioButton, 至少配备一个 ButtonGroup。



GUI 设计提示

当用户只能从一组选项中选择惟一的一个选项时, 可以考虑使用 JRadioButton 组件。



GUI 设计提示

总是将属于同一组的 JRadioButton 放置在一个独立的 ButtonGroup 中。

一旦有某个 JRadioButton 被选取 (内含一个黑色圆点), 其 isSelected 方法将返回 boolean 类型的一个 true 值。如果某个 JRadioButton 未被选取 (为空), 那么, 其方法 isSelected 将返回一个 false 值。

当 JRadioButton 被选取时, 同样也会产生一个(ActionEvent) 事件。因此, 当 JRadioButton 被选取时, 将调用 actionPerformed 事件处理程序。

以下伪代码描述了成绩评定应用程序的基本操作:


```
当用户点击 Submit Grades JButton 时
    从 JTextField 中取得该名学生的姓名和成绩
    将学生信息添加至相应的数组中
    在 JTextArea 中显示每位学生的姓名、测验成绩及平均分
    在 Class average: JTextField 中显示班级平均分
    将该学生的姓名和成绩从相应的 JTextField 中删除

如果已输入 10 名学生
    禁用 Submit Grades JButton

当用户选取 Numeric JRadioButton 时
    在 JTextArea 中显示每位学生的姓名、数值型测验成绩以及数值型平均分
    在 Class average: JTextField 中显示班级的数值型平均分

当用户选取 Letter JRadioButton 时
    在 JTextArea 中显示每位学生的姓名、字母型测验成绩以及字母型平均分
    在 Class average: JTextField 中显示班级的字母型平均分
```

该成绩评定应用程序, 将在用户选择是以字母成绩显示还是以数值成绩显示时, 利用 JRadioButton 组件的 actionPerformed 事件处理程序, 完成 JTextArea 中文本内容的更新。既然已探试过了成绩评定应用程序并研究了它的伪代码表示, 下面, 将使用一张 ACE 表, 帮助读者最终把这个伪代码转换成 Java 的实现。图 17.6 中列出了该应用程序中相应的操作、组件以及事件, 以帮助读者完成属于自己版本的这一应用程序。



操作	组件	事件
标记应用程序中的组件	studentNameJLabel test1JLabel test2JLabel test3JLabel displayJLabel classAverageJLabel inputGradeJPanel submitGradesJButton	当应用程序运行时
从 JTextField 中 取得该名学生的姓名和成绩	studentNameJTextField test1JTextField, test2JTextField, test3JTextField	当用户点击 Submit Grades JButton 时
将学生信息添加至 相应的数组中	studentNames studentGrades	

(续表)

操作	组件	事件
在 JTextArea 中显示每位学生的姓名 测验成绩以及平均分	displayJTextArea	
在 Class average: JTextField 中显示 班级平均分	classAverageJTextField	
将该学生的姓名和成绩 从相应的 JTextField 中删除	studentNameJTextField test1TextJLabel, test2TextJLabel, test3TextJLabel	
如果已输入 10 名学生 禁用 Submit Grades JButton	submitGradesJButton	
在 JTextArea 中显示每位学生的姓名、 数值型测试成绩以及数值型平均分	numericJRadioButton displayJTextArea	当用户选取 Numeric JRadioButton 时
在 Class average: JTextField 中显示 班级的数值型平均分	classAverageJTextField	
在 JTextArea 中显示每位学生的姓名、 字母型测试成绩以及字母型平均分	letterJRadioButton displayJTextArea	当用户选取 Letter JRadioButton 时
在 Class average: JTextField 中显示 班级的字母型平均分	classAverageJTextField	

图 17.6 成绩评定应用程序的 ACE 表

接下来，将创建自己的成绩评定应用程序并首次使用本书中所讲授的 JRadioButton 组件。利用这样的 一个 JRadioButton，使用户可以以字母形式或者是数值形式来查看学生的成绩。

在应用程序中使用 JRadioButton

- 1. 将模板复制到工作目录中 将 C:\Examples\Tutorial17\TemplateApplication\StudentGrades 目录复制到 C:\SimplyJava 目录中。
- 2. 打开成绩评定应用程序的模板文件 在自己的文本编辑器中打开模板文件 StudentGrades.java。
- 3. 自定义 numericJRadioButton 将图 17.7 中第 166 行至第 168 行添加到程序代码中。第 166 行通过使用 JRadioButton 的 setBounds 方法，将这一 numericJRadioButton 的范围设置为 55, 244, 75, 23。第 167 行通过使用 numericJRadioButton 的 setText 方法，将字符串 "Numeric" 显示在 numericJRadioButton 的右侧。第 168 行则通过 numericJRadioButton 的 setSelected 方法使其处于被选取状态（为 true 值）。JRadioButton 在创建时，默认为未被选取的状态。



错误预防提示

为避免潜在的逻辑错误，通过设置 selected 属性的 true 值，将一组 JRadioButton 中的一个作为默认情况下的选取状态。

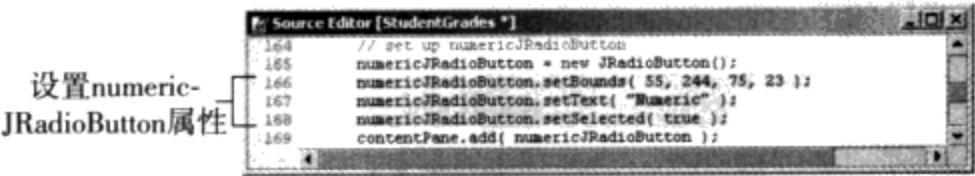


图 17.7 自定义 numericJRadioButton

- 4. 自定义 letterJRadioButton 将图 17.8 中第 187 行至第 188 行添加到程序代码中。第 187 行是将 letterJRadioButton 的范围设置为 140, 244, 75, 23。第 188 行则是将 letterJRadioButton 的文本设置为 "Letter"。
- 5. 将 numericJRadioButton 加入 ButtonGroup 将图 17.9 中第 169 行添加到程序代码中。此行通过使用 ButtonGroup 的 add 方法，将这一 numericJRadioButton 添加到 displayButtonGroup 中。此时，若 displayButtonGroup 中另有其他 JRadioButton 被选取时，相应的 numericJRadioButton 将撤销选取。

设置letterJRadioButton的属性

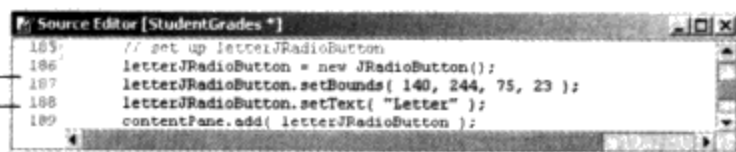


图 17.8 自定义 letterJRadioButton

通过调用ButtonGroup
的add方法加入某个
JRadioButton

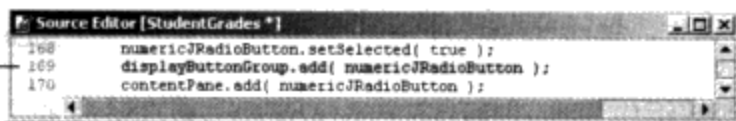


图 17.9 将 numericJRadioButton 加入 ButtonGroup



GUI 设计提示

对同一组中的 JRadioButton 按照水平或垂直方式对齐。

6. 将 letterJRadioButton 加入 ButtonGroup 将图 17.10 中第 190 行添加到程序代码中。此行同样也是将 letterJRadioButton 添加到 displayButtonGroup 中。

调用ButtonGroup的add方法
加入另一个JRadioButton

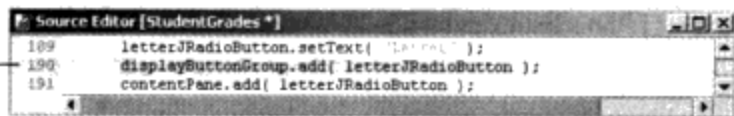


图 17.10 将 letterJRadioButton 加入 ButtonGroup

7. 保存应用程序 保存修改后的源代码文件。
8. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\StudentGrades 进入到当前工作目录中。
9. 编译应用程序 通过键入 javac StudentGrades.java 编译该应用程序。
10. 运行应用程序 若此应用程序能够正确编译，通过键入 java StudentGrades 来运行它。图 17.11 中显示了更新后的应用程序的运行结果。现在，我们可输入一位学生的姓名、测验成绩，然后，选取其中一个 JRadioButtons，当点击 Submit JButton 以后，平均分并未计算。

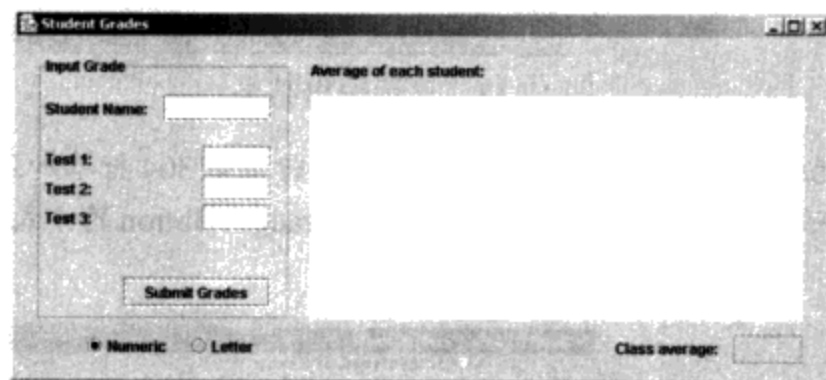


图 17.11 运行已添加 JRadioButton 的应用程序

11. 关闭正在运行的应用程序 点击关闭按钮关闭正在运行的应用程序。
12. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

自测题

1. 哪一个方法可用来确定 JRadioButton 的状态?
a) isSelected b) getSelected c) isChecked d) getChecked
2. 当某个 JRadioButton 被选取时，会相应地调用 _____ 事件处理程序。
a) checkedChanged b) actionPerformed c) selectedChanged d) 以上答案都不对

答案：1) a 2) b

17.4 在成绩评定应用程序中添加代码

到目前为止，我们已在应用程序中放置了所需要的组件。下面，将准备编写相应的代码，与用户提供的数据实现交互。首先，将完成方法 submitGradesJButtonActionPerformed。

完成方法 submitGradesJButtonActionPerformed

- 1. 将学生添加到数组中 将图 17.12 中第 276 行至第 282 行添加到程序代码中。第 277 行是将该名学生的姓名添加到一维数组 studentNames 中。第 278 行至第 280 行是将该名学生的三门测验成绩添加到二维数组 studentGrades 中。注意，这些数组已在第 60 行和第 63 行上得到了创建。第 282 行是对班级的学生人数做自增运算。

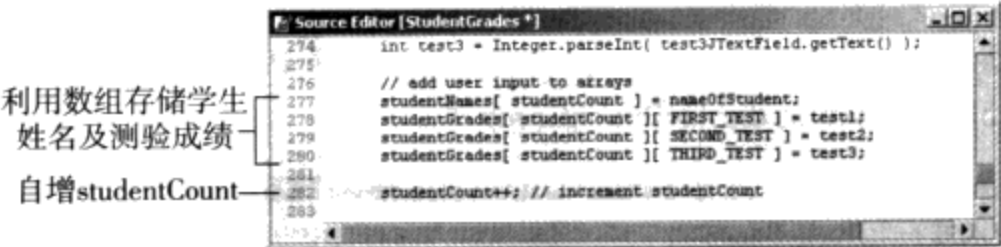


图 17.12 存储输入

- 2. 显示输出 将图 17.13 中第 284 行至第 291 行添加到程序代码中。第 284 行通过使用 JRadioButton 的 isSelected 方法，确定用户所希望显示的学生成绩方式。第 286 行调用了一个 displayNumericGrades 方法，第 290 行同样也调用了一个相类似的 displayLetterGrades 方法。这些方法将在随后的内容中进行定义。

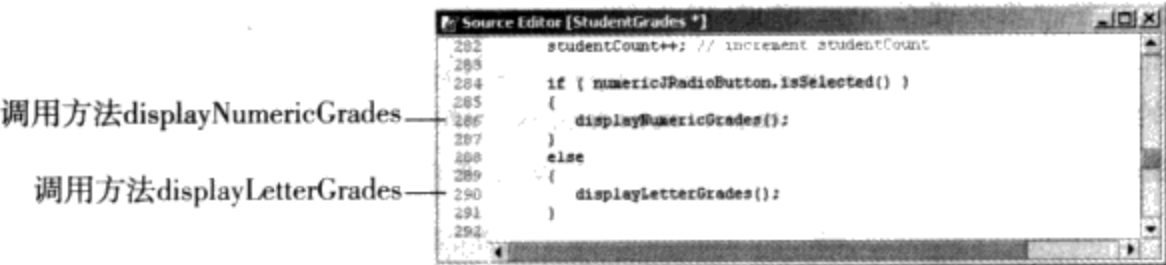


图 17.13 显示输出结果

- 3. 禁用 Submit Grades JButton 将图 17.14 中第 299 行至第 304 行添加到程序代码中。如果已有 10 名学生加入数组，那么第 303 行将通过禁用 Submit Grades JButton 的方式，使之不再接收更多的学生成绩。

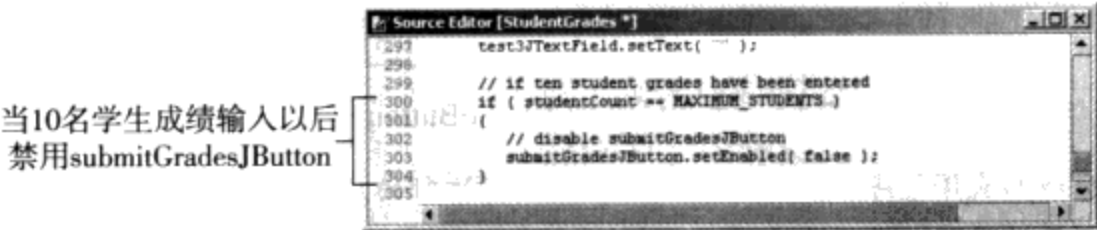


图 17.14 应用程序不允许有超过 10 项的数据输入

- 4. 保存应用程序 保存修改后的源代码文件。

在上面这个示例中，虽然调用了 displayNumericGrades 方法和 displayLetterGrades 方法，但这些用于显示适当数据的方法实际上还未得到定义。下面，将首先定义一个 displayNumericGrades 方法。

定义显示数值成绩的方法

- 1. 定义 displayNumericGrades 方法 将图 17.15 中第 308 行至第 318 行添加至 submitGradesJButtonActionPerformed 方法的后面。这些代码行将用于定义方法 displayNumericGrades，此方法以数值类型的

形式显示成绩。第 312 行至第 313 行是向 displayJTextArea 中添加一个表头。第 315 行至第 316 行，分别声明并初始化了两个变量，用来存储学生总成绩和班级总成绩。

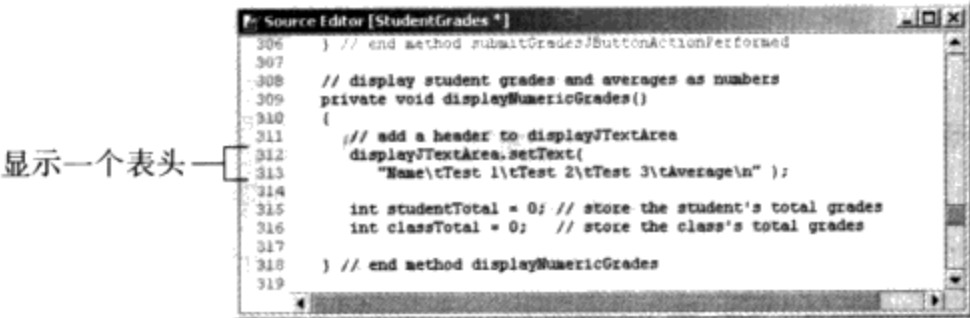


图 17.15 在 displayJTextArea 中添加一个表头

2. 输出学生姓名 将图 17.16 中第 318 行至第 325 行添加到程序代码中。其中，for 语句将对已添加至该班级的每位学生进行迭代。第 321 行是把学生的姓名追加到 displayJTextArea 中。第 323 行初始化了一个变量，用来存储该名学生的总成绩。此变量将把班级内每名学生的总成绩初始化为 0。

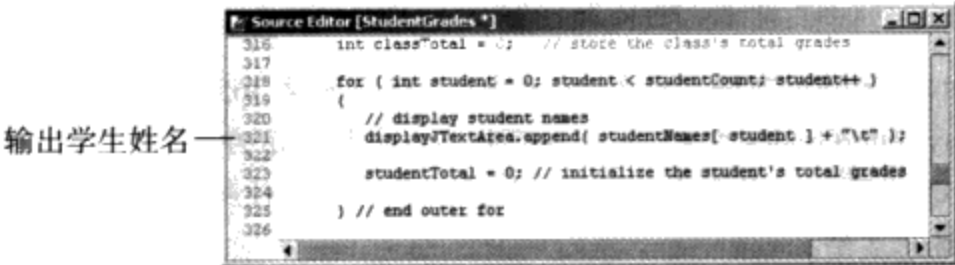


图 17.16 将学生姓名追加至 displayJTextArea 中

3. 输出学生的成绩 将图 17.17 中第 325 行至第 334 行添加到上一步骤中 for 语句的语句体内。新添加的这个 for 语句被称为嵌套循环，因为它封装（嵌套）在另一个控制语句的内部（即上一步骤中 for 语句的语句体内）。嵌套循环常常用来处理二维数组。这条 for 语句采用步进方式对 studentGrades 数组中的每次测验成绩进行处理。第 328 行至第 329 行是把每一次测验的成绩追加至 displayJTextArea 的后面。第 332 行是把测验成绩累加到变量 studentTotal 中。

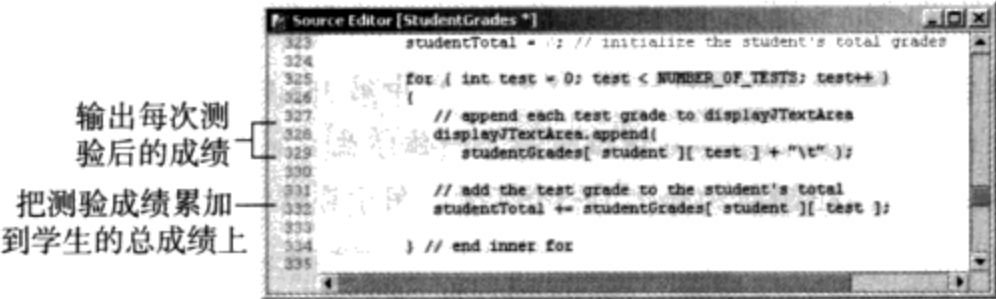


图 17.17 输出每次测验的成绩并计算学生的总成绩

4. 输出学生的平均成绩 将图 17.18 中第 336 行至第 343 行添加到程序代码中。第 337 行将把每名学生的总成绩累加至班级总成绩中。第 340 行至第 341 行计算该名学生的平均成绩，第 342 行至第 343 行则把该名学生的平均成绩追加到 displayJTextArea 中。在第 67 行中声明的实例变量 twoDigits，将其格式化得到的结果。

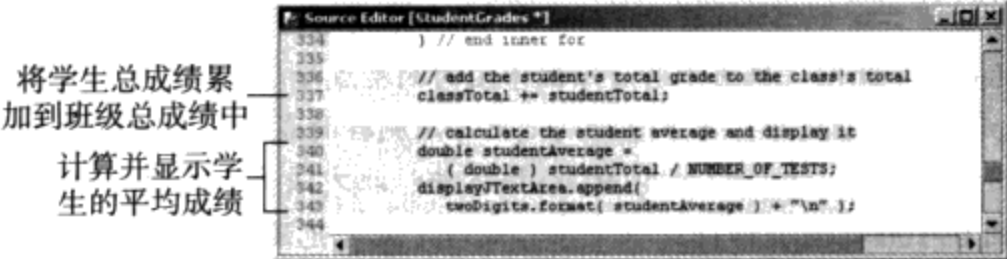


图 17.18 计算并显示学生的平均成绩

5. 输出班级平均成绩 将图 17.19 中第 347 行至第 351 行添加到程序代码中。第 348 行至第 349 行将计算班级的平均测验成绩。第 350 行至第 351 行是将该班级的平均成绩输出至 classAverageJTextField 中。

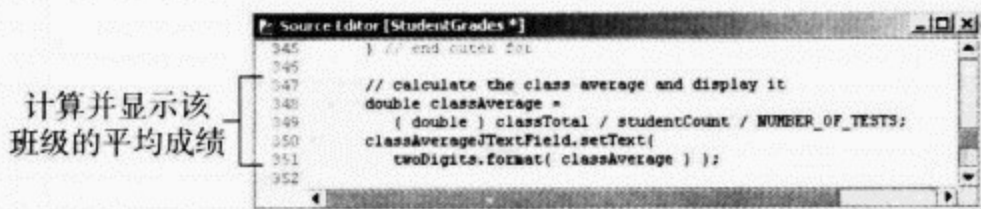


图 17.19 计算并显示该班级的平均成绩

6. 保存应用程序 保存修改后的源代码文件。

现在，我们通过定义 displayNumericGrades 方法，将学生成绩、学生平均成绩以及班级的平均成绩以数值形式显示在了 displayJTextArea 中。下面，定义另一个方法 displayLetterGrades，将同样的信息以字母成绩的形式进行输出。

定义显示字母成绩的方法

1. 定义 displayLetterGrades 方法 将图 17.20 中第 355 行至第 365 行添加到程序代码中。这些代码行将用于定义 displayLetterGrades 方法，该方法可把学生成绩显示为字母型成绩。第 359 行至第 360 行是向 displayJTextArea 中添加一个表头。第 362 行至第 363 行分别声明并初始化了两个变量，用来存储学生总成绩和班级总成绩。

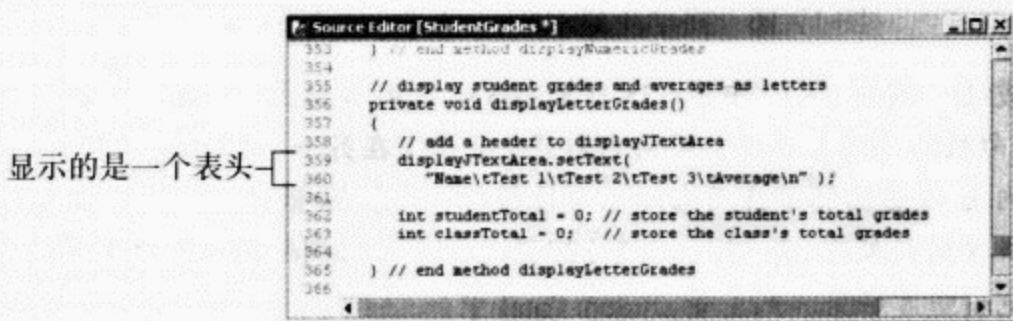


图 17.20 输出显示一个表头

2. 输出学生的姓名并初始化 studentTotal 将图 17.21 中第 365 行至第 372 行添加到程序代码中。第 365 行开始的 for 语句将遍历已添加到该班级的每一位学生。第 368 行则是把每位学生的姓名追加至 displayJTextArea 中。第 370 行初始化了一个变量，用于存储学生的总成绩。

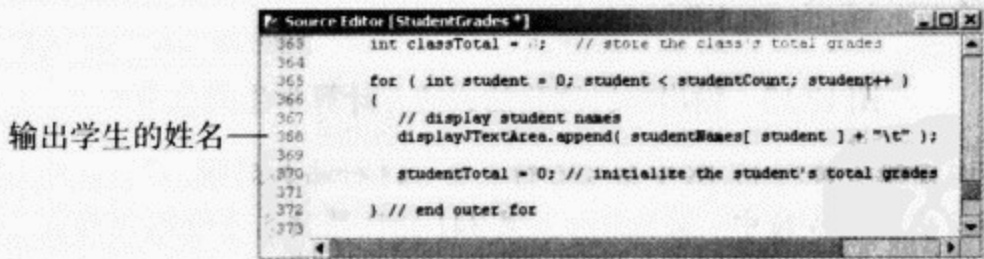


图 17.21 输出每位学生的姓名

3. 输出学生的成绩 将图 17.22 中第 372 行至第 381 行添加到上一步骤中 for 语句的语句体内，从而得到一个嵌套的循环。此段代码用于遍历每位学生的测验成绩。第 375 行至第 376 行是将当前的测验成绩追加至 displayJTextArea 中。第 379 行则是把每次测验的成绩累加到该学生的总成绩当中。注意，第 375 行调用的是一个 convertToLetterGrade 方法。此方法（已为读者提供）能够将学生的数值成绩（处于范围 0~100 之间）转换成一个字母成绩（处于范围 A~F 之间）。
4. 输出学生的平均成绩 将图 17.23 中第 383 行至第 390 行添加到程序代码中。第 387 行至第 388 行，用

于计算学生的平均成绩。第 389 行至第 390 行，是把学生的平均成绩追加至 displayJTextArea 中。注意，第 390 行再次调用了 convertToLetterGrade 方法。

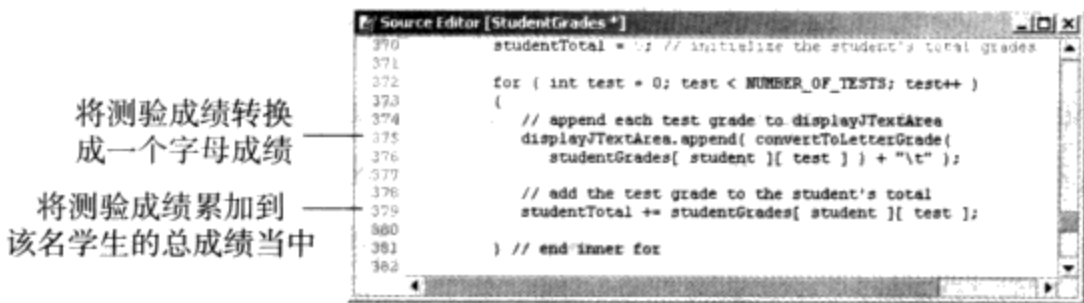


图 17.22 输出学生的字母成绩并计算学生的总成绩

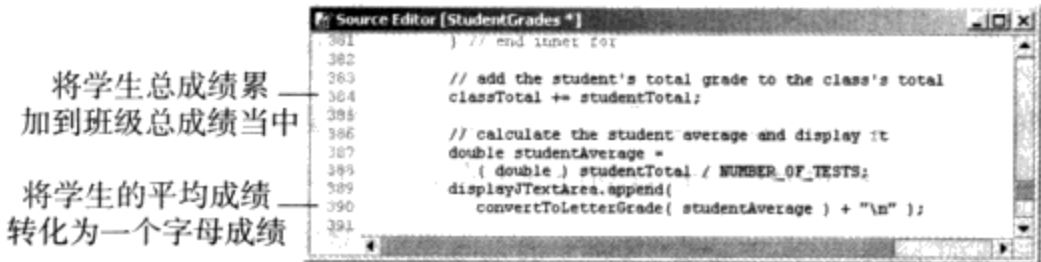


图 17.23 计算并显示学生的平均字母成绩

5. 输出班级的平均成绩 将图 17.24 中第 394 行至第 398 行添加到程序代码中。第 395 行至第 396 行为计算该班级的平均成绩。第 397 行至第 398 行是将该班级的平均成绩输出到 classAverageJTextField 中。第 398 行再次调用了方法 convertToLetterGrade。

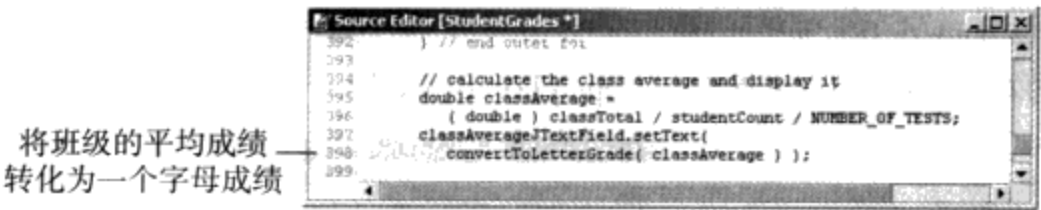


图 17.24 计算并显示该班级的字母型平均成绩

- 6. 保存应用程序 保存修改后的源代码文件。
- 7. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\StudentGrades 进入到当前工作目录中。
- 8. 编译应用程序 通过键入 javac StudentGrades.java 编译该应用程序。
- 9. 运行应用程序 若此应用程序能够正确编译，通过键入 java StudentGrades 来运行它。图 17.25 中显示了更新后的应用程序的运行结果。在 Student Name: JTextField 中输入 Gretta Green。在 “Test1:”, “Test2:” 和 “Test3:” JTextField 中分别输入 87, 94 和 93。之后，点击 Submit Grades JButton。可以看到，相对应的数值成绩被立刻显示在了 JTextArea 中。注意，此时若选取 Letter JRadioButton，将不会产生任何作用，因为现在还未添加该项功能。

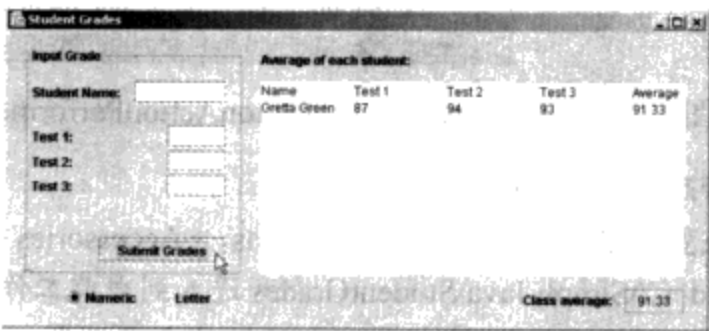


图 17.25 运行已定义了新方法的应用程序

- 10. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- 11. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

下面,准备编写一个事件处理程序,作为改进该应用程序的功能,即允许用户将所得结果显示为字母型成绩或者是数值型成绩。

编写 JRadioButton 的事件处理程序

1. 补充完成 numericJRadioButton 的 actionPerformed 事件处理程序 将图 17.26 中第 179 行添加到程序代码中。方法 numericJRadioButtonActionPerformed 是在 Numeric JRadioButton 被选取时开始调用的。

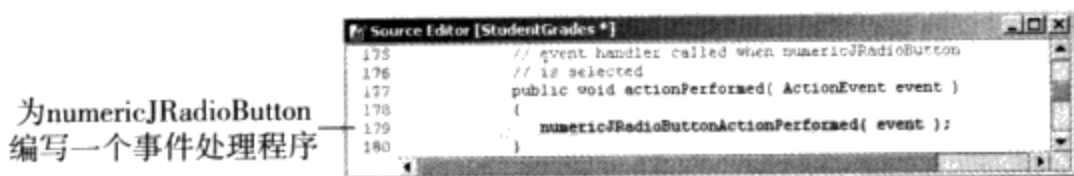


图 17.26 numericJRadioButton 的 actionPerformed 事件处理程序

2. 补充完成 letterJRadioButton 的 actionPerformed 事件处理程序 将图 17.27 中第 200 行添加到程序代码中。方法 letterJRadioButtonActionPerformed 是在 Letter JRadioButton 被选取时调用的。

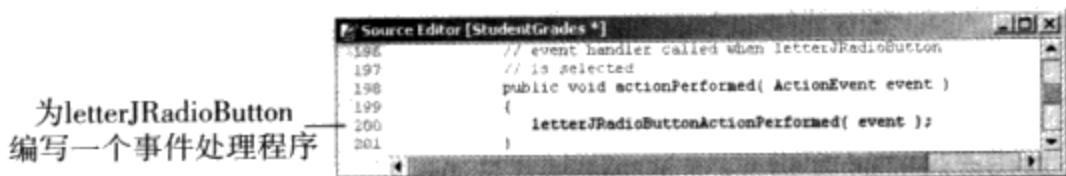


图 17.27 letterJRadioButton 的 actionPerformed 事件处理程序

3. 定义方法 numericJRadioButtonActionPerformed 将图 17.28 中第 402 行至第 408 行添加到程序代码中,从而定义方法 numericJRadioButtonActionPerformed。第 406 行将调用方法 displayNumericGrades。该方法是把数据以数值成绩的形式显示在 displayJTextArea 中。

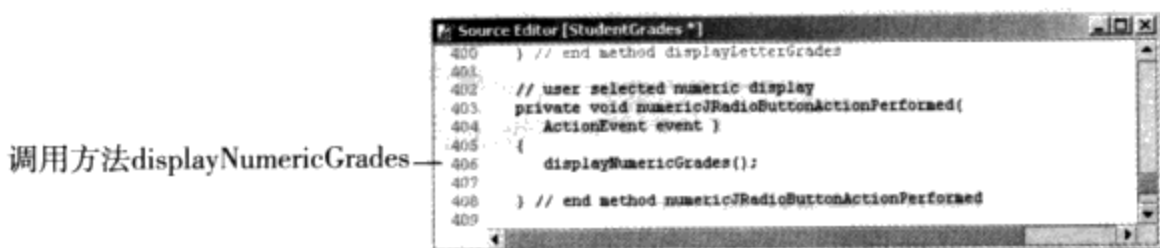


图 17.28 方法 numericJRadioButtonActionPerformed

4. 定义方法 letterJRadioButtonActionPerformed 将图 17.29 中第 410 行至第 416 行添加到程序代码中并调用相应的方法 displayLetterGrades。此方法则是把数据以字母成绩的形式显示在 displayJTextArea 中。

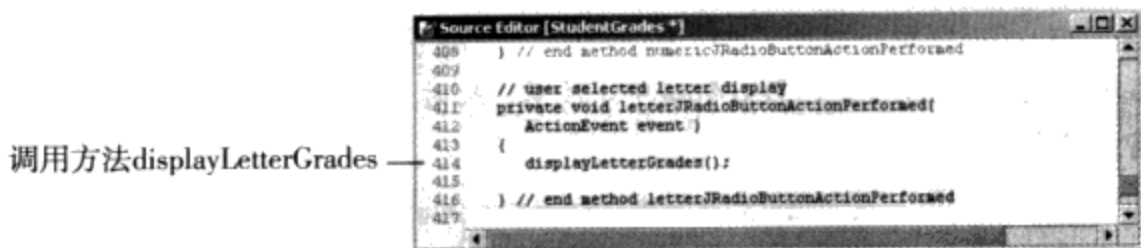


图 17.29 方法 letterJRadioButtonActionPerformed

5. 保存应用程序 保存修改后的源代码文件。
6. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\StudentGrades 进入到当前工作目录中。
7. 编译应用程序 通过键入 javac StudentGrades.java 编译该应用程序。
8. 运行应用程序 若此应用程序能够正确编译,通过键入 java StudentGrades 来运行它。此时,可按照字母或数值型的显示方式,查看所显示的成绩。

9. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。

10. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

图 17.30 中给出了成绩评定应用程序的完整源代码。在本教程中, 凡需要添加、查看或者是修改的代码, 均在图中相应的代码行中进行了突出显示。

```
1 // Tutorial 17: StudentGrades.java
2 // This application computes each student's grade average and
3 // the class average for ten students.
4 import java.awt.*;
5 import java.awt.event.*;
6 import java.text.*;
7 import javax.swing.*;
8 import javax.swing.border.*;
9
10 public class StudentGrades extends JFrame
11 {
12     // JPanel for user inputs
13     private JPanel inputGradeJPanel;
14
15     // JLabel and JTextField for student name
16     private JLabel studentNameJLabel;
17     private JTextField studentNameJTextField;
18
19     // JLabel and JTextField for test 1 score
20     private JLabel test1JLabel;
21     private JTextField test1JTextField;
22
23     // JLabel and JTextField for test 2 score
24     private JLabel test2JLabel;
25     private JTextField test2JTextField;
26
27     // JLabel and JTextField for test 3 score
28     private JLabel test3JLabel;
29     private JTextField test3JTextField;
30
31     // JButton to calculate student and class average
32     private JButton submitGradesJButton;
33
34     // ButtonGroup to control numeric and letter JRadioButtons
35     private ButtonGroup displayButtonGroup;
36
37     // JRadioButtons to choose to display numerically or as letters
38     private JRadioButton numericJRadioButton;
39     private JRadioButton letterJRadioButton;
40
41     // JLabel, JTextArea and JScrollPane to display students averages
42     private JLabel displayJLabel;
43     private JTextArea displayJTextArea;
44
45     // JLabel and JTextField to display the class average
46     private JLabel classAverageJLabel;
47     private JTextField classAverageJTextField;
48
49     // initialize number of students to zero
50     private int studentCount = 0 ;
51
52     // constants
53     private final int NUMBER_OF_TESTS = 3 ;
```

```
54 private final int MAXIMUM_STUDENTS = 10 ;
55 private final int FIRST_TEST = 0 ;
56 private final int SECOND_TEST = 1 ;
57 private final int THIRD_TEST = 2 ;
58
59 // one-dimensional array to store student names          创建用于存储学生
60 private String studentNames[] = new String[ MAXIMUM_STUDENTS ]; 姓名的数组
61
62 // two-dimensional array to store student grades          创建用于存储学生成绩的数组
63 private int studentGrades[][] =
64     new int [ MAXIMUM_STUDENTS ][ NUMBER_OF_TESTS ];
65
66 // DecimalFormat for two digits of precision
67 private DecimalFormat twoDigits = new DecimalFormat( "0.00" );
68
69 // no-argument constructor
70 public StudentGrades()
71 {
72     createUserInterface();
73 }
74
75 // create and position GUI components; register event handlers
76 private void createUserInterface()
77 {
78     // get content pane for attaching GUI components
79     Container contentPane = getContentPane();
80
81     // enable explicit positioning of GUI components
82     contentPane.setLayout( null );
83
84     // set up inputGradeJPanel
85     inputGradeJPanel = new JPanel();
86     inputGradeJPanel.setBounds( 16 , 16 , 208 , 218 );
87     inputGradeJPanel.setBorder(
88         new TitledBorder( "Input Grade" ) );
89     inputGradeJPanel.setLayout( null );
90     contentPane.add( inputGradeJPanel );
91
92     // set up studentNameJLabel
93     studentNameJLabel = new JLabel();
94     studentNameJLabel.setBounds( 8 , 32 , 90 , 23 );
95     studentNameJLabel.setText( "Student Name:" );
96     inputGradeJPanel.add( studentNameJLabel );
97
98     // set up studentNameJTextField
99     studentNameJTextField = new JTextField();
100    studentNameJTextField.setBounds( 104 , 32 , 88 , 21 );
101    studentNameJTextField.setHorizontalAlignment(
102        JTextField.RIGHT );
103    inputGradeJPanel.add( studentNameJTextField );
104
105    // set up test1JLabel
106    test1JLabel = new JLabel();
107    test1JLabel.setBounds( 8 , 74 , 60 , 23 );
108    test1JLabel.setText( "Test 1:" );
109    inputGradeJPanel.add( test1JLabel );
110
111    // set up test1JTextField
112    test1JTextField = new JTextField();
113    test1JTextField.setBounds( 136 , 74 , 56 , 21 );
114    test1JTextField.setHorizontalAlignment( JTextField.RIGHT );
```

```

176         // is selected
177         public void actionPerformed((ActionEvent event)
178         {
179             numericJRadioButtonActionPerformed( event );
180         }
181         // end anonymous inner class
182     }; // end call to addActionListener
183
184     // set up letterJRadioButton
185     letterJRadioButton = new JRadioButton();
186     letterJRadioButton.setBounds( 140 , 244, 75 , 23 );
187     letterJRadioButton.setText( "Letter" );
188     displayButtonGroup.add( letterJRadioButton );
189     contentPane.add( letterJRadioButton );
190     letterJRadioButton.addActionListener(
191         new ActionListener() // anonymous inner class
192         {
193             // event handler called when letterJRadioButton
194             // is selected
195             public void actionPerformed( ActionEvent event )
196             {
197                 letterJRadioButtonActionPerformed( event );
198             }
199         }
200     );
201     // end anonymous inner class
202 }; // end call to addActionListener
203
204 // set up displayJLabel
205 displayJLabel = new JLabel();
206 displayJLabel.setBounds( 240 , 16 , 150 , 23 );
207 displayJLabel.setText( "Average of each student:" );
208 contentPane.add( displayJLabel );
209
210 // set up displayJTextArea
211 displayJTextArea = new JTextArea();
212 displayJTextArea.setBounds( 240 , 48, 402, 184 );
213 displayJTextArea.setEditable( false );
214 contentPane.add( displayJTextArea );
215
216 // set up classAverageJLabel
217 classAverageJLabel = new JLabel();
218 classAverageJLabel.setBounds( 490 , 244, 96 , 23 );
219 classAverageJLabel.setText( "Class average:" );
220 contentPane.add( classAverageJLabel );
221
222 // set up classAverageJTextField
223 classAverageJTextField = new JTextField();
224 classAverageJTextField.setBounds( 586 , 244, 56 , 23 );
225 classAverageJTextField.setHorizontalAlignment(
226     JTextField.CENTER );
227 classAverageJTextField.setEditable( false );
228 contentPane.add( classAverageJTextField );
229
230 // set properties of application's window
231 setTitle( "Student Grades" ); // set title bar string
232 setSize( 670, 308 ); // set window size
233 setVisible( true ); // display window

```

为 numericJRadioButton 编写事件处理程序

设置

letterJRadioButton 的属性

将 letterJRadioButton 加入 displayButtonGroup

为 letterJRadioButton 编写事件处理程序

```

237
238 } // end method createUserInterface
239
240 // convert a number to a letter grade
241 private String convertToLetterGrade( double grade )
242 {
243     if ( grade >= 90 )
244     {
245         return "A" ;
246     }
247     else if ( grade >= 80 )
248     {
249         return "B" ;
250     }
251     else if ( grade >= 70 )
252     {
253         return "C" ;
254     }
255     else if ( grade >= 60 )
256     {
257         return "D" ;
258     }
259     else
260     {
261         return "F" ;
262     }
263 } // end method convertToLetterGrade
264
265 // calculate and display the student and class average
266 private void submitGradesJButtonActionPerformed(
267     ActionEvent event )
268 {
269     // get user input
270     String nameOfStudent = studentNameJTextField.getText();
271     int test1 = Integer.parseInt( test1JTextField.getText() );
272     int test2 = Integer.parseInt( test2JTextField.getText() );
273     int test3 = Integer.parseInt( test3JTextField.getText() );
274
275     // add user input to arrays
276     studentNames[ studentCount ] = nameOfStudent;
277     studentGrades[ studentCount ][ FIRST_TEST ] = test1;
278     studentGrades[ studentCount ][ SECOND_TEST ] = test2;
279     studentGrades[ studentCount ][ THIRD_TEST ] = test3;
280
281     studentCount++; // increment studentCount
282
283     if ( numericJRadioButton.isSelected() )
284     {
285         displayNumericGrades();
286     }
287     else
288     {
289         displayLetterGrades();
290     }
291
292     // clear other JTextFields for new data
293     studentNameJTextField.setText( "" );
294     test1JTextField.setText( "" );
295     test2JTextField.setText( "" );
296

```

将学生姓名和测验
成绩存储到数组中

自增 studentCount

调用方法 displayNumericGrades

调用方法 displayLetterGrades


```

355 // display student grades and averages as letters
356 private void displayLetterGrades()
357 {
358     // add a header to displayJTextArea
359     displayJTextArea.setText(
360         "Name\tTest 1\tTest 2\tTest 3\tAverage\n" );      显示一个表头
361
362     int studentTotal = 0 ; // store the student's total grades
363     int classTotal = 0 ; // store the class's total grades
364
365     for ( int student = 0 ; student < studentCount; student++ )
366     {
367         // display student names
368         displayJTextArea.append( studentNames[ student ] + "\t" );      输出学生姓名
369
370         studentTotal = 0 ; // initialize the student's total grades
371
372         for ( int test = 0 ; test < NUMBER_OF_TESTS; test++ )
373         {
374             // append each test grade to displayJTextArea
375             displayJTextArea.append( convertToLetterGrade(
376                 studentGrades[ student ][ test ] ) + "\t" );      将测验成绩转化为字母型成绩
377
378             // add the test grade to the student's total
379             studentTotal += studentGrades[ student ][ test ];      将测验成绩累加到该学生的总成绩当中
380
381         } // end inner for
382
383         // add the student's total grade to the class's total
384         classTotal += studentTotal;      将学生的总成绩累加到班级总成绩当中
385
386         // calculate the student average and display it
387         double studentAverage =
388             ( double ) studentTotal / NUMBER_OF_TESTS;      将学生的平均成绩
389         displayJTextArea.append(
390             convertToLetterGrade( studentAverage ) + "\n" );      转化为字母型成绩
391
392     } // end outer for
393
394     // calculate the class average and display it
395     double classAverage =
396         ( double ) classTotal / studentCount / NUMBER_OF_TESTS;      将班级平均
397     classAverageJTextField.setText(
398         convertToLetterGrade( classAverage ) );      成绩转化为字母型成绩
399
400 } // end method displayLetterGrades
401
402 // user selected numeric display
403 private void numericJRadioButtonActionPerformed(
404    (ActionEvent event )
405 {
406     displayNumericGrades();      调用方法 displayNumericGrades
407
408 } // end method numericJRadioButtonActionPerformed
409
410 // user selected letter display
411 private void letterJRadioButtonActionPerformed(
412    (ActionEvent event )

```

```
413 {  
414     displayLetterGrades();  
415  
416 } // end method letterJRadioButtonActionPerformed  
417  
418 // main method  
419 public static void main( String[] args )  
420 {  
421     StudentGrades application = new StudentGrades();  
422     application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );  
423  
424 } // end method main  
425  
426 } // end class StudentGrades
```

图 17.30 成绩评定应用程序代码

自测题

1. 一个 ButtonGroup 中可包含 _____ JRadioButton。
a) 两个 b) 不多于一个 c) 不多于三个 d) 任意数量
2. 当某 ButtonGroup 中有一个 JRadioButton 被选取时，_____。
a) 还可同时选取其他的 JRadioButton b) 还可同时选取两个 JRadioButton
c) 其他所有已选取的 JRadioButton 将被取消 d) (a)和(c)

答案：1) d 2) c

17.5 小结

在本教程中，学习了如何使用二维数组存储数据，以及如何声明并赋值二维数组。同时，通过使用嵌套循环，对一个二维数组中存储的数据进行了相应的处理。

为了帮助读者更好地完成成绩评定应用程序，介绍了有关 JRadioButton 组件的使用。通过学习，使读者明白必须要将相关的 JRadioButton 组织在一个 ButtonGroup 中。而最初的情况是，每个 ButtonGroup 中可以没有或者只有惟一的一个 JRadioButton 处于被选取状态。一旦有 JRadioButton 被选取，相应的 ButtonGroup 将确保任何时候只能有惟一的一个 JRadioButton 处于被选取的状态。当 ButtonGroup 中再选取另外一个 JRadioButton 时，则该 ButtonGroup 中前一个已被选取的 JRadioButton 会被取消。

在学完有关 JRadioButton 的知识以后，通过编写一些代码，将用户输入的数据存储到了一个二维数组当中。与此同时，还了解到，通过选取某个 JRadioButton 会相应地调用一个 actionPerformed 事件处理程序。

到目前为止，我们一直都在使用一些类，从表示应用程序 GUI 的 JFrame 类到用来生成随机数的 Random 类。在下一个教程中，将学习如何在应用程序中创建属于自己的类。

技术小结

使用二维数组

- 通过声明一个二维数组，可创建一个由值组成的表（每一行将包含相同数目的列）。例如，使用代码：

```
int numbers[][] = new int [ 3 ][ 2 ];
```

来声明一个 3 行 2 列的数组。

使用 JRadioButton

- 通过在应用程序中使用 JRadioButton，可允许用户从一组选项中选择惟一的一个选项。

选取 JRadioButton

- 点击 JRadioButton 上的空白小圆圈。该圆圈内将出现一个黑色小圆点。

判定 JRadioButton 是否已被选取

- 利用方法 isSelected 判定 JRadioButton 的状态。此方法将在 JRadioButton 处于被选取状态时返回 true，反之则返回 false。

当 JRadioButton 处于被选取状态时执行相应的代码

- 当 JRadioButton 被选取时，可执行 actionPerformed 事件处理程序。

关键术语

ButtonGroup 的 add 方法 用于将 JRadioButton 添加到某个 ButtonGroup 组中。

ButtonGroup 由任意数目的 JRadioButton 所构成的一个分组。任何时刻，ButtonGroup 中只能有惟一的一个 JRadioButton 处于被选取状态。当另一个 JRadioButton 被选取时，前面所选取的 JRadioButton 将被取消。

列 在访问某二维数组中的一个元素时，第二个索引代表的便是列。

JRadioButton 的 isSelected 方法 若该 JRadioButton 已被选取，则返回 true，反之返回 false。

JRadioButton 组件 该组件在外观上是一个小圆圈，其内部可为空（当未经选取时）或者包含一个黑色的圆点（当被选取时）。通常需要将两个或更多数量的这种组件组织在一个分组中。

m × n 数组 一个 m 行 n 列的二维数组。

嵌套循环 一个被封装在另一个控制语句体内的循环（如一条 for 语句）。

行 在访问某二维数组中的一个元素时，第一个索引代表的便是行。

JRadioButton 的 setSelected 方法 可将 JRadioButton 的状态设置为选取状态(true)或者是未选取(false) 状态。

状态按钮 一种只能处于“开”或“关”状态的按钮（如 JRadioButton 和 JCheckBox ）。

表 以行和列的形式来组织信息的一个二维数组。

二维数组 需要通过两个索引标识某项值的数组。这些数组通常用来表示其值以行和列的形式进行安排的信息表。

GUI 设计指导

JRadioButton

- 当用户只能从一组选项中选择出惟一的一个选项时，可考虑使用 JRadioButton 组件。
- 总是将属于同一组的 JRadioButton 放置在一个独立的 ButtonGroup 中。
- 对同一组中的 JRadioButton 按照水平或垂直方式对齐。

Java 类库索引

JRadioButton 该组件允许用户在多个选项中只选取一个选项。

- 运行



- 事件处理程序

actionPerformed 当 JRadioButton 组件被选取时会得到调用。

- 方法

isSelected 若 JRadioButton 已被选取，则返回 true，反之则返回 false。

setBounds 指定 JRadioButton 组件的位置 (相对其控制容器左上角的位置), 包括组件的高度和宽度 (以像素为单位)。

setSelected 将某个 JRadioButton 的状态设置为已选状态 (true) 或未选 (false) 状态。当为 true 时, JRadioButton 将显示为一个内含黑色圆点的小圆圈。当为 false 时, JRadioButton 将显示为一个空白小圆圈。

setText 指定 JRadioButton 中的文本显示。

ButtonGroup 此类将多个 JRadioButton 组件组织在一个分组中。初始情况下, 可以没有或者只有一个 JRadioButton 处于被选取状态。如果一个 JRadioButton 已被选取, 则任何时刻相应 ButtonGroup 中只能有惟一的一个 JRadioButton 处于被选取状态。

● 方法

add 向 ButtonGroup 中添加一个 JRadioButton。

习题

选择题

- 17.1 使用 _____ 方法可对 JRadioButton 组件进行选取或取消。
a) setSelected b) setChecked c) setDefault d) setEnabled
- 17.2 组件 _____ 是一种状态组件。
a) JRadioButton b) JCheckBox c) JButton d) (a)和(b)
- 17.3 在一个 $m \times n$ 的数组中, m 代表 _____。
a) 该数组的列数 b) 数组元素的总个数 c) 该数组的行数 d) 每一行元素数目
- 17.4 语句 _____ 将把一个 5 行 3 列的数组赋值给一个 int 型二维数组变量 myArray。
a) myArray = new int[5][3]; b) myArray = new int[4][2];
c) myArray = new int[3][5]; d) myArray = new int[2][4];
- 17.5 JRadioButton 是一种 _____ 类型的按钮。
a) 复选 b) 变化 c) 状态 d) 操作
- 17.6 通过使用一个 _____ 可将多个 JRadioButton 组织到相应的 JFrame 中。
a) GroupBox b) ButtonBox c) ButtonGroup d) 以上答案都不对
- 17.7 通过使用 _____ 方法可将一个 JRadioButton 添加到某个 ButtonGroup 组中。
a) add b) addJButton c) addJRadioButton d) newJRadioButton
- 17.8 二维数组通常用来表示 _____。
a) 饼分图 b) 位距 c) 线条 d) 表
- 17.9 当创建一个 JRadioButton 时, 它 _____。
a) 处于已选取状态 c) 处于未选取状态
c) 已被添加到内容面板中 d) 已被添加到一个默认的 ButtonGroup 中
- 17.10 创建一个 5 行 5 列的 int 型二维数组 myArray 的正确操作是 _____。
a) int myArray[][] = new int[5][5] b) int myArray[2] = new int(5 , 5)
c) int myArray[,] = new int[5 , 5] d) int myArray[] = new int[5 , 5]

练习题

- 17.11 (食谱问卷调查应用程序) 某学校食堂为改善午餐种类, 向学生进行了一次电子问卷调查。创建这样的一个应用程序, 通过使用二维数组存储有关调查的计数统计 (参见图 17.31)。同时, 还需为参加调查的学生提供一个 JRadioButton 组件, 利用该组件表明他们是否喜欢某一特定食品。
- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial17\Exercises\FoodSurvey 目录复制到 C:\SimplyJava 目录中。
- b) 打开模板文件。在自己的文本编辑器中打开 FoodSurvey.java 文件。

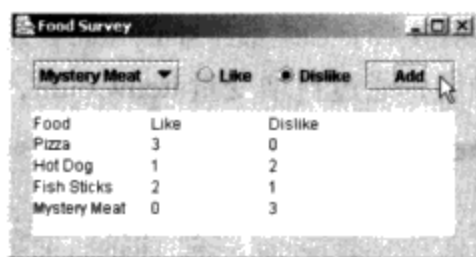


图 17.31 食谱问卷调查应用程序

- c) **声明一个 int 型二维数组** 在第 29 行至第 30 行, 插入相应的代码, 声明一个名为 display 的 4 行 2 列的 int 型二维数组。
 - d) **声明局部变量** 在第 105 行上的方法 addJButtonActionPerformed 的内部, 将 displayJTextArea 的文本设置为 "Food\\Like\\Dislike", 使其作为一个表头。创建一个 int 型局部变量 index。利用该变量存储 foodJComboBox 中所选择项目的一个索引。
 - e) **使用 for 循环显示数据** 插入一条 for 语句, 利用其循环通过 foodChoices 数组中的每一行 (0~3)。在该循环的循环体内, 将正确的食品名称追加至 displayJTextArea 中。而 for 语句中的计数器变量将作为 foodChoices 数组的一个索引。在 displayJTextArea 中, 为所显示的食品名称前添加转义序列 "\n", 同样, 在食品名称的后面再添加一个转义序列 "\t"。
 - f) **判定用于自增的计数器** 在步骤(e)中所创建的 for 语句的语句体内插入一条 if 语句。该语句将用于检查 likeJRadioButton 是否已被选取, 以及变量 index 是否等于 for 语句的计数器。如果两个条件都为 true, 自增 display 数组第 0 列的计数器。插入 else 语句, 其中包含一条嵌套的 if 语句, 判断 dislikeJRadioButton 是否已被选取及变量 index 是否等于该 for 语句的计数器。如果两个条件都为 true, 自增 display 数组第 1 列的计数器。
 - g) **将 display 数组中的内容添加到输出中** 使用一条嵌套的 for 语句, 将 display 数组中的内容追加至 displayJTextArea 中。将第一条 for 语句中的计数器作为 display 的第一个索引 (代表行号), 而将嵌套 for 语句中的计数器作为 display 的第二个索引 (代表列号)。在所添加的每个数组元素的后面, 再加入一个转义序列 "\n"。
 - h) **保存应用程序** 保存修改后的源代码文件。
 - i) **打开命令提示符窗口改变目录** 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\FoodSurvey 进入到当前工作目录中。
 - j) **编译应用程序** 通过键入 javac FoodSurvey.java 编译该应用程序。
 - k) **运行完成后的应用程序** 若能正确编译应用程序, 键入 java FoodSurvey 来运行它。选择任意的一个 Like 或 Dislike JRadioButton。然后, 点击 Add JButton, 确信 JTextArea 中所有的字符串及其数值都能正确进行显示。向 Food Survey 加入其他的一些选项, 确保数值的正确性。
 - l) **关闭应用程序** 点击关闭按钮关闭正在运行的应用程序。
 - m) **关闭命令提示符窗口** 点击命令提示符窗口的关闭按钮将其关闭。
- 17.12 (销售报告应用程序) 某服装制造商, 要求开发一个用于计算一周内销售总额的应用程序。其中, 每款服装的销售情况应单独输入, 但应对该服装在 5 个工作日内的所有销售情况一次输入。应用程序应计算出一周内每款服装的销售总额, 以及该制造商所有服装的销售总额。由于企业规模不大, 因而, 在任何工作周内, 该企业最多只能生产出 10 款服装。应用程序的运行结果应参照图 17.32 中的所示。
- a) **将模板复制到工作目录中** 将 C:\Examples\Tutorial17\Exercises\SalesReport 目录复制到 C:\SimplyJava 目录中。
 - b) **打开模板文件** 在自己的文本编辑器中打开 SalesReport.java 文件。
 - c) **输入数据** 从第 231 行开始, 添加相应的代码, 实现用户数据的输入。变量 nameOfItem 将用于存储每款服装的名称, 并利用索引 itemCount (存储所添服装种类的数目) 将其赋予数组 itemNames。变量 monday, tuesday, wednesday, thursday 和 friday 将分别存储对应工作日内的

销售数据。这些变量必须被赋予二维数组 `dailyItems`。该数组的第一个索引为 `itemCount`，第二个索引为 0~4 之间。最后，通过自增变量 `itemCount`，表示另一款服装的销售数据已被添加。

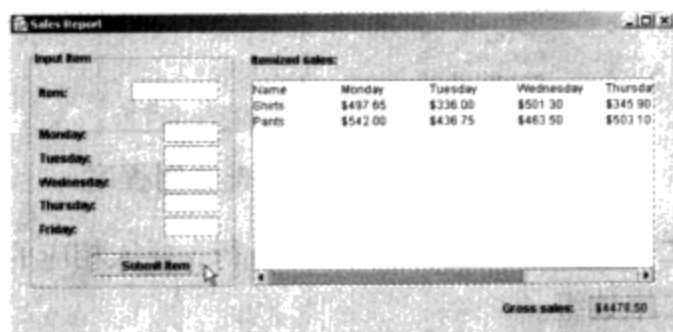


图 17.32 销售报告应用程序

- d) 遍历所有已添加的服装 在 `displaySales` 方法的内部，位于声明变量 `salesTotal` 语句（参见第270行）的后面，添加实现一条 `for` 语句的代码。使这样一条 `for` 语句能够从 0 遍历至 `itemCount`。声明变量 `item`，作为 `for` 语句的计数器。
 - e) 显示服装的名称 在这条 `for` 语句的内部，插入相应的代码，将每款服装的名称追加并显示在 `displayJTextArea` 中。切记，相应名称已被存储在 `String` 型数组 `itemNames` 中。通过追加转义序列 `"\n"`，正确格式化所输出的结果。
 - f) 遍历工作周 添加代码，将变量 `weekTotal` 初始化为 0。此变量将记录每款服装一周之内的销售总额。再添加一条 `for` 语句。该 `for` 语句将从 0 遍历至一周内的工作天数（存储在模板内一个已声明的常量 `NUMBER_OF_DAYS` 中）。
 - g) 追加显示每日销售额并求出一周内的销售总额 在这条 `for` 语句的内部添加相应的代码，将每日销售额追加并显示在 `displayJTextArea` 中。这些销售额已存储在数组 `dailyItems` 中。此数组利用当前正在计算的一款服装，以及某一个工作日进行访问。由于输出结果应为货币值，因此，需使用 `DecimalFormat` 的变量 `dollars`，对这一结果值进行格式化。同时，通过追加转义序列 `"\n"`，正确格式化输出的结果。
 - h) 计算周销售额 插入相应的代码，将日销售额累加到变量 `weekTotal` 中。该变量将存储每款服装的周销售额。最后，通过添加一个右花括号，结束步骤(f)中所创建的 `for` 语句。
 - i) 计算总销售额并输出周销售额 插入相应代码，将周销售额累加到变量 `salesTotal` 中。该变量用以记录一周内全部服装的销售总额。添加相应的代码，将周销售额追加并显示在 `displayJTextArea` 中。由于周销售额将以货币的形式进行存储，因此，还需再一次使用 `DecimalFormat` 的对象 `dollars`。同样，通过追加一个转义序列 `"\n"`，将 `JTextArea` 中的输出推到一行上。最后，通过添加一个右花括号，结束步骤(d)中创建的 `for` 语句。
 - j) 保存应用程序 保存修改后的源代码文件。
 - k) 打开命令提示符窗口改变目录 选择 `Start` → `Programs` → `Accessories` → `Command Prompt` 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\SalesReport` 进入到当前工作目录中。
 - l) 编译应用程序 通过键入 `javac SalesReport.java` 编译该应用程序。
 - m) 运行完成后的应用程序 若能正确编译应用程序，键入 `java SalesReport` 来运行它。
 - n) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
 - o) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。
- 17.13 (利润报告应用程序) 销售报告应用程序给服装制造商留下了极其深刻的印象（参见习题 17.12），因此，他们又要求开发一个利润报告应用程序。该应用程序同销售报告应用程序类似，所不同的是，还允许输入有关获益或损耗的信息。应用程序将通过提供相应的 `JRadioButton`，让用户为某个特定领域选择是获益还是损耗（参见图 17.33）。
- a) 将模板复制到工作目录中 将 `C:\Examples\Tutorial17\Exercises\ProfitReport` 目录复制到 `C:\SimplyJava` 目录中。

- b) 打开模板文件 在自己的文本编辑器中打开 ProfitReport.java 文件。
- c) 修改模板应用程序 参照习题 17.12 修改此模板应用程序。
- d) 自定义 Gain JRadioButton 在方法 createUserInterface 中添加用以自定义 gainJRadioButton 的代码。参照图 17.33 中出现的组件, 为其设置相应的范围属性及文本属性。将该 JRadioButton 设置成当应用程序启动时, 已得到选取的状态 (默认状态)。然后, 将这个 JRadioButton 加入到 profitButtonGroup 组中。
- e) 自定义 Loss JRadioButton 在方法 createUserInterface 中添加用以自定义 lossJRadioButton 的代码。参照图 17.33 中出现的组件, 为其设置相应的范围属性及文本属性。然后, 将这个 JRadioButton 也加入到 profitButtonGroup 组中。

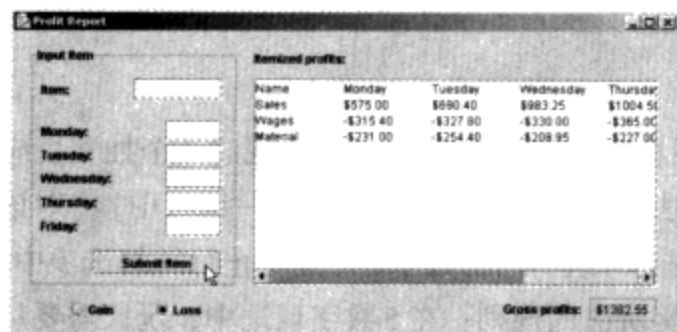


图 17.33 利润报告应用程序

- f) 对已选取的 JRadioButton 进行测试 在方法 submitItemJButtonActionPerformed 中添加相关的代码, 测试两个 JRadioButton 中的哪一个已被选取。如果 Gain JRadioButton 被选取, 则将输入值按照正常数值累加到数组中。如果 Loss JRadioButton 被选取, 则应把输入值以负数值的形式累加到数组中。
 - g) 保存应用程序 保存修改后的源代码文件。
 - h) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\ProfitReport 进入到当前工作目录中。
 - i) 编译应用程序 通过键入 javac ProfitReport.java 编译该应用程序。
 - j) 运行完成后的应用程序 若能正确编译应用程序, 键入 java ProfitReport 来运行它。
 - k) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
 - l) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。
- 17.14 (说出这段代码的作用) 下列代码的返回结果是什么? 假定 getStockPrices 是一个可返回某个 2×31 数组的方法。其中, 第一行为某月内各交易日开盘时的股票价格, 第二行为各交易日结束时的股票价格。

```

1 private int [] mystery()
2 {
3     int [][] prices = new int [ 2 ][ 31 ];
4     prices = getStockPrices();
5     int [] result = new int [ 31 ];
6     for ( int i = 0 ; i <= 30; i++ )
7     {
8         result[ i ] = prices[ 0 ][ i ] - prices[ 1 ][ i ];
9     } // end for
10    return result;
11 } // end method mystery

```


17.15 (找出代码中的错误) 寻找下列代码中的错误。方法 twoDArrays 可创建出一个二维数组, 其中的所有值将被初始化为 1。

```

1 private void twoDArrays()
2 {
3     int [][] intArray;
4
5     intArray = new int [ 4 ][ 4 ];
6
7     // assign 1 to all cell values
8     for ( int i = 0 ; i < 4 ; i++ )
9     {
10         intArray[ i ][ i ] = 1 ;
11
12     } // end for
13
14 } // end method twoDArrays

```

挑战题

17.16 (改进的抽奖机应用程序) 在教程 15 中, 曾出现过一个抽奖机应用程序, 该应用程序能够从 4 种不同类型的抽奖玩法中选择出一些数字。在这一练习中, 将对该抽奖机应用程序进行改进, 使之可针对 5 数字玩法选择 4 组不同的数字, 并能防止每一组选项中出现相同的数字 (参见图 17.34)。首先回顾一下该抽奖玩法的规则: 在 5 数字玩法中, 玩家需要从 0~39 之间选出 5 个惟一的数字。

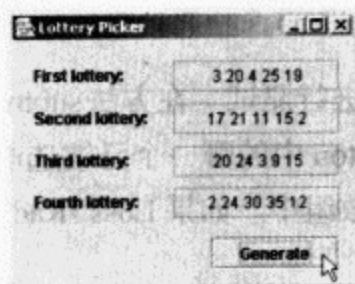


图 17.34 改进的抽奖机应用程序

- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial17\Exercises\EnhancedLotteryPicker 目录复制到 C:\SimplyJava 目录中。
- b) 打开模板文件 在自己的文本编辑器中打开 LotteryPicker.java 文件。
- c) 迭代 4 轮抽奖 在应用程序中第 138 行, 添加相应代码, 实现一条 for 语句, 并使其循环执行 4 次——每一次循环将代表一轮抽奖。使用变量 lottery 作为这一 for 语句的计数器。
- d) 初始化 boolean 型数组 为了在每轮抽奖中生成一些惟一的数字, 将使用一个 boolean 类型的二维数组 uniqueNumber (已在模板中生成)。当一轮抽奖过程中的一个数字被选取时, 会将数组 (通过该轮次的抽奖数目及所选的数字进行索引) 中相应变量的值设置为 true。首先, 必须将该轮抽奖的数字值初始化为 false。通过添加一个嵌套的 for 语句, 将该轮抽奖数组中的 40 个值设置为 false。需要在这个 for 语句中使用 lottery 和计数器对数组来进行索引。最后, 通过添加了一个右花括号来结束这个内嵌的 for 语句。
- e) 初始化存储多个选项的字符串 作为输出的数组在模板中已提供给了读者。它是一个具有 4 个值的字符串数组——每个值代表一轮抽奖。该数组将存储每轮抽奖过程中所生成的数字结果。添加相应的代码, 将存储抽奖结果的字符串初始化成一个空的字符串 ("")。需要使用 lottery 来索引这一数组。
- f) 迭代 5 个被选数字 每轮抽奖过程中所产生的 5 个惟一数字都需要得到选取。添加另一条用以实现 5 次迭代的 for 语句。这一 for 语句在每次执行的时候, 都会生成一个惟一的抽奖数字。
- g) 生成惟一的抽奖数字 在这条 for 语句的内部添加一条 do...while 语句。该 do...while 语句的语句体将使用到方法 generate (已在模板中声明)。为此方法传递参数 0 和 39, 使其能够返回一个 0~39 (含 0 和 39) 的随机数, 为该值赋予变量 selection (已在模板中声明)。该循环的继

续条件为测试 selection 中所存储的数字, 判断是否已在该轮抽奖过程中使用过。通过使用由 lottery 和 selection 索引的数组 uniqueNumber 来实现。

- h) **更新 uniqueNumber 中的值** 添加相应的代码, 把由所选数字表示的 uniqueNumber (通过 lottery 和 selection 进行索引) 中的值设置为 true。这时, 该数字在此轮抽奖中将不会再被选中。
- i) **将数字添加至输出中** 插入相应的代码, 将一个空格和所选择的数字添加到存储此轮抽奖结果的字符串中。该字符串的结果将存储在由 lottery 索引的 output 数组中。之后, 添加一个右花括号, 结束这一内嵌的 for 语句 [由步骤(f)创建]。然后, 再添加另外一个花括号, 结束那个外部 for 语句 [由步骤(c)创建]。
- j) **显示生成的数字** 数组 output 将把所生成的数字序列存储在它的 4 个索引中。添加相应的代码, 将每个 JTextField (oneJTextField, twoJTextField, threeJTextField 以及 fourJTextField) 的文本设置为数组 output 中的一个字符串值。
- k) **保存应用程序** 保存修改后的源代码文件。
- l) **打开命令提示符窗口改变目录** 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\EnhancedLotteryPicker` 进入到当前工作目录中。
- m) **编译应用程序** 通过键入 `javac LotteryPicker.java` 编译该应用程序。
- n) **运行完成后的应用程序** 若能正确编译应用程序, 键入 `java LotteryPicker` 来运行它。点击 Generate JButton, 检查 4 组抽奖数字中是否每一组中的所有数字都是惟一的。通过多次测验, 确认其结果是否正确。
- o) **关闭应用程序** 点击关闭按钮关闭正在运行的应用程序。
- p) **关闭命令提示符窗口** 点击命令提示符窗口的关闭按钮将其关闭。

教程 18 微波炉模拟应用程序

创建属于自己的类及其对象



教学目标

在本教程中，读者将学到以下内容：

- 声明一个属于自己的类
- 创建并使用属于自己类的对象
- 控制实例变量的访问
- 关键字 `private` 的使用
- `get` 方法与 `set` 方法的定义

在前面的教程中，我们一直使用如下方法开发应用程序：通过分析所要构建的典型问题，从 Java 类库中确定出用以实现应用程序所需要的类，然后，选择这些类中的适当方法或者是创建其他一些方法来最终完成该应用程序的开发。

我们已见过 Java 类库中的许多类。例如，每个 GUI 组件都是作为一个类被声明的。当把某个组件添加到应用程序中时，需创建一个该类的对象（也称实例）并将其添加到相应的应用程序中。一个类可以有多个实例；例如，在本书中见到的许多应用程序中都包含了两个以上的 `JButton`。与此同时，还看到过 Java 类库中一些不属于 GUI 组件的类。例如，`DecimalFormat` 类，该类可用来创建 `DecimalFormat` 对象（处理文本数据）。当应用程序创建并使用某类的对象时，该应用程序被认为是此类的一个客户程序。

在本教程中，将学习如何创建并使用自己的类（有时也称程序员定义的类）。创建自己的类是面向对象程序设计（OOP，Object-Oriented Programming）中的一个关键部分，这是因为类可以被多个应用程序复用。在 Java 程序设计的世界里，应用程序的创建是将 Java 类库中的类及其方法，同程序员定义的类及其方法相互组合而得到的。我们已经在本书中创建过一些方法，从本教程开始，将学习如何创建包含这些方法的类。

将要创建的应用程序是一个允许用户输入烹调时间的微波炉模拟程序。为处理时间数据，需要创建一个称之为 `CookingTime` 的类。此类用于存储分钟数及秒数（微波炉模拟应用程序通过它们记录剩余的烹调时间）。为使客户应用程序能够访问和改变分钟数及秒数，还将为该提供相应的 `get` 方法和 `set` 方法。

18.1 探试微波炉模拟应用程序

当在本教程中创建微波炉模拟应用程序时，需要构建一个属于自己的类。该应用程序需满足以下需求：

应用程序需求分析

某电子设备公司正考虑投产微波炉。该公司现要求开发一个可模拟微波炉工作原理的应用程序。这个微波炉模拟应用程序将包含一个允许用户设定微波炉烹调时间的小键盘,相应的烹调时间也应显示给用户。一旦输入某个时间,用户便可通过点击 Start JButton 来启动整个烹调过程。此时,微波炉上的玻璃窗将改变颜色(从灰色变为黄色),从而模拟食物烹调过程中微波炉内的灯光颜色,相应的定时器也将按每秒一次的速度进行递减。当时间终止时,该微波炉的玻璃窗会再次返回到灰色状态(表示微波炉已停止工作),然后,显示文本 "Done!"。用户可在任何时刻通过点击 Clear JButton 停止微波炉的工作,然后再重新输入一个新的时间。注意,用户所输入的分钟数不能超过 59,秒数也不能超过 59;否则,任何无效的烹调时间都将被重置为零。

我们将以这个完成后的应用程序的探试作为起点,之后,学习另外的一些 Java 技术以最终创建一个属于自己的应用程序。



探试微波炉模拟应用程序

- 1. 定位到完成后的应用程序 选择 Start→Programs→Accessories→Command Prompt 打开一个命令提示符窗口。键入 cd C:\Examples\Tutorial18\CompletedApplication\MicrowaveOven 将目录改变到这个完成后的微波炉模拟应用程序的目录下面。
- 2. 运行微波炉模拟应用程序 在命令提示符窗口下键入 java MicrowaveOven 运行该应用程序(参见图 18.1)。该应用程序 GUI 的左侧为一个面积较大的矩形(代表微波炉的玻璃窗),右侧则是一个小键盘,内含了一个其文本内容为 Microwave Oven 的 JTextField。数字 JButton 将用来输入烹调时间,输入的时间将最终显示在面板右上侧的一个 JTextField 中。

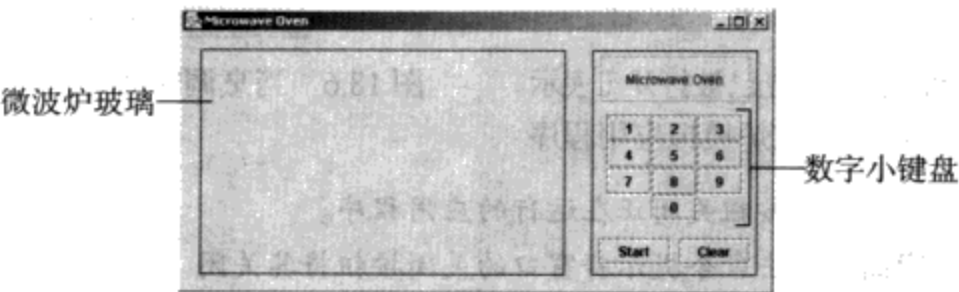


图 18.1 微波炉模拟应用程序执行时的 GUI 显示

- 3. 输入一个时间 按照 1, 2, 3, 4, 5 的顺序依次点击相应的数字 JButton。注意,所输入的数字不能超过四位(前两位代表分钟,后两位代表秒)——否则,任何额外的数字位都不会显示出来(参见图 18.2)。分钟数和秒数必须小于或等于 59。若用户输入的是一个无效分钟数或者秒数的话(比如 89),该无效数据将被重置为零。

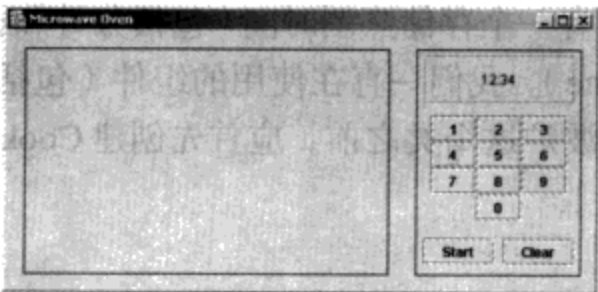


图 18.2 微波炉模拟应用程序只能接受四位数字

- 4. 输入无效数据 点击 Clear JButton 清除原来的输入。按照 7, 2, 3, 5 的顺序依次点击相应的数字 JButton (参见图 18.3)。因为分钟数 72 超过了所允许的最大值 59,因而此输入属于无效输入。点击 Start JButton

时，该分钟数将被重置为零。可以看到，其中所显示的分钟数的确被重置成了00，而秒数仍保持不变（参见图18.4）。与此同时，注意观察该微波炉窗口已变为黄色，说明模拟微波炉内部工作的灯光已经打开，这样，用户便可查看到所烹调的食物。

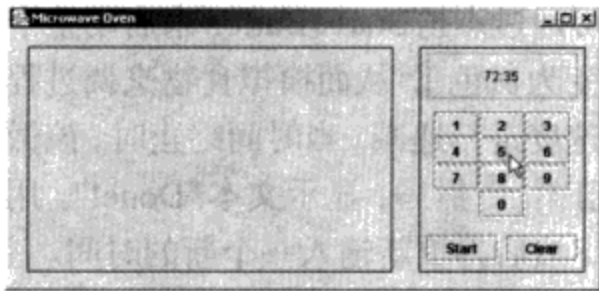


图 18.3 带有无效输入的微波炉模拟应用程序

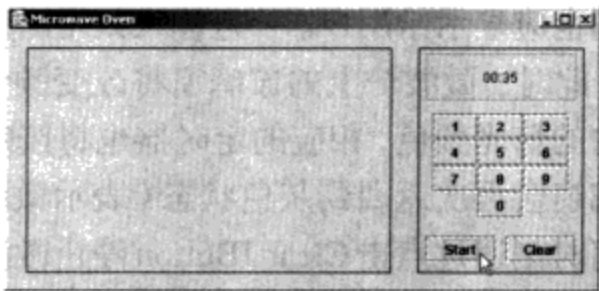


图 18.4 输入无效数据并点击 Start JButton 以后的微波炉模拟应用程序

- 5. 输入有效数据 点击 Clear JButton 以便重新输入一个新的烹调时间。点击 5 JButton（代表 5 秒）；之后，点击 Start JButton（参见图 18.5）。
- 6. 当烹调时间终止时查看该应用程序 等待 5 秒钟。注意观察显示时间的 JTextField，其中的时间将每隔 1 秒进行递减。当时间递减至零时，用于显示的 JTextField 内的文本将变为 "Done!"，而微波炉窗口的颜色也将变回为原来 JFrame 的颜色，以此表明该微波炉的灯光已关闭（参见图 18.6）。

黄色模拟微波炉内的灯光

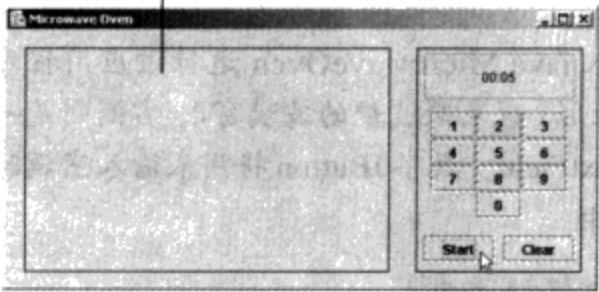


图 18.5 当输入有效时间且内部灯光已被打开（表示正在烹调食物）时的微波炉模拟应用程序

当烹调结束时
变回默认颜色从而
JTextField中将显示"Done!" 模拟烹调已经结束

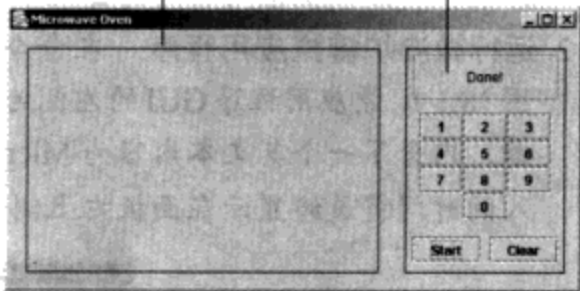


图 18.6 当烹调时间终止时的微波炉模拟应用程序

- 7. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- 8. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

18.2 设计微波炉模拟应用程序

微波炉模拟应用程序通过使用 JPanel 组件将其内部的一些组件组织了起来。该应用程序需要两个 JPanel——一个组织该应用程序中的 JButton 和 JTextField，另一个则代表微波炉的玻璃窗。相应的模板应用程序已提供了其中的一个 JPanel，我们只需添加另外的一个 JPanel。

微波炉模拟应用程序中含有一个存储烹调时间（包括分钟和秒）的对象，此对象出自一个烹调时间类（类名为 CookingTime）。我们一直在使用的组件（包括 JFrame 组件），均出自 Java 类库中已定义好的类。在创建微波炉这个类之前，应首先创建 CookingTime 类。下列伪代码描述了 CookingTime 类的基本操作：

- 当创建时间对象时
 - 将分钟数和秒数设置为 0
- 当设置分钟时
 - 如果分钟数大于 0 且小于 60


```
    将分钟数设置为指定值
  否则
    将分钟数设置为 0

  当设置秒数时
    如果秒数大于 0 且小于 60
      将秒数设置为指定值
    否则
      将秒数设置为 0

  当时间对象递减时
    如果秒数大于 0
      将秒数减 1
    否则如果分钟数大于 0
      将分钟数减 1
      将秒数设置为 59
```

当一个 `CookingTime` 类的对象被创建时，其分钟数和秒数将同时被初始化为 0。任何无效的分钟数或者秒数数据都会使其值被重置为 0。以下伪代码描述了微波炉模拟应用程序的基本操作：

```
  当用户点击一个数字 JButton 时
    显示格式化的时间

  当用户点击 Start JButton 时
    显示格式化的时间
    存储分钟数及秒数
    开启倒计时
    打开微波炉的灯光

  当定时器时间间隔结束时（时间为 1 秒钟）
    每隔 1 秒将时间减 1

    如果新的时间不为 0
      显示该时间
    否则
      停止倒计时
      显示文本 "Done!"
      将微波炉的灯光关闭

  当用户点击 Clear JButton 时
    停止倒计时
    显示文本 "Microwave Oven"
    关闭微波炉内的灯光
```

用户是通过点击数字 JButton 来完成输入的。每当一个数字 JButton 被点击时，该 JButton 上的数字将被追加至 GUI 中用于显示烹调时间的那个 JTextField 的末尾。在输入完烹调时间以后，可通过点击 Start JButton 启动烹调过程，或者是点击 Clear JButton 重新输入一轮新的时间。如果点击的是 Start JButton，那么将利用 Timer 组件开启倒计时功能，同时微波炉窗口也将变为黄色，说明该微波炉的灯光已经打开（方便用户观察所烹调的食物）。每隔一秒钟，会更新剩余烹调时间的显示。当倒计时结束时，会在 `displayJTextField` 中显示出文本 "Done!"，而微波炉内的“灯光”也将通过窗口变为默认的灰色，表明微波炉已经关闭。

在开始构建自己的应用程序之前，有必要先创建一张关于该应用程序的 ACE 表。图 18.7 中列出了完成该微波炉模拟应用程序所需的操作、组件及其事件。


操作	组件 / 对象	事件	
	oneJButton, twoJButton threeJButton, fourJButton fiveJButton, sixJButton sevenJButton eightJButton, nineJButton	当用户点击某个数字 JButton 时	
显示格式化的时间	displayJTextField	当用户点击 Start JButton 时	
显示格式化的时间	startJButton		
存储分钟数及秒数	displayJTextField microwaveTime (CookingTime)		
开启倒计时	clockTimer(Timer)		
打开微波炉的灯光	windowJPanel		
每隔 1 秒将时间减 1	clockTimer(Timer) microwaveTime (CookingTime)	当计时器时间间隔结束时	
如果新的时间不为 0 显示该时间 否则	microwaveTime, displayJTextField clockTimer(Timer)	当用户点击 Clear JButton 时	
停止倒计时	displayJTextField windowJPanel clearJButton		
显示文本 "Done!"			
将微波炉的灯光关闭			
停止倒计时			
显示文本 "Microwave Oven"	clockTimer(Timer) displayJTextField		
关闭微波炉内的灯光	windowJPanel		

图 18.7 微波炉模拟应用程序的 ACE 表

当点击其中任何一个数字 JButton 时, 都会将输入发送至应用程序中, 而输入的数字值也将显示在 displayJTextField 中。一旦输入所有数据, 便可通过点击 Start JButton 开启倒计时。windowJPanel 的背景色将被设置为黄色, 从而模拟已经打开的微波炉灯光, 而 clockTimer 将在倒计时的过程中, 以每隔 1 秒的速度更新 displayJTextField 中的内容。为了能清除原先的输入并重新启动微波炉, 可以通过点击 Clear JButton 来实现。

下面, 将学习如何在应用程序中添加一个类, 并应用该类创建一个包含分钟数和秒数的对象。

创建一个类

1. 将模板复制到工作目录中 将 C:\Examples\Tutorial18\TemplateApplication\MicrowaveOven 目录复制到 C:\SimplyJava 目录中。
2. 打开 CookingTime 类的模板文件 在自己的文本编辑器中打开模板文件 CookingTime.java。切记, 在 Java 中每一个源代码文件的文件名应与包含 public 类的类名相同, 并后跟一个 .java 的文件扩展名。文件 MicrowaveOven.java 和 CookingTime.java 中将分别包含两个类 MicrowaveOven 和 CookingTime。这两个类是创建微波炉模拟应用程序所必需的。
3. 查看模板代码 观察图 18.8 中第 1 行至第 7 行。第 1 行至第 2 行为一个表示类文件名称及其作用的注释。第 4 行是定义 CookingTime 类的起始行, 此行包含关键字 public 和 class, 之后是该类的类名 (这里为 CookingTime)。关键字 class 表明之后是一个类的声明。本书中所有的类都将声明为 public, 这是创建由多个应用程序复用相关类的第一步。一个类只有被声明为 public 并放置在其他应用程序所能导入的包中时, 才是可被复用的。有关创建包的知识不属于本书所讲述的范围。左右花括号 (位于第 5 行和第 7 行) 分别代表该类类体的开始和结束。在某类的类体中声明的方法或变量被称为类的成员。



好的编程习惯

按照惯例, 类名以一个大写字母作为开始, 其后相继单词中的第一个字母也应该大写。

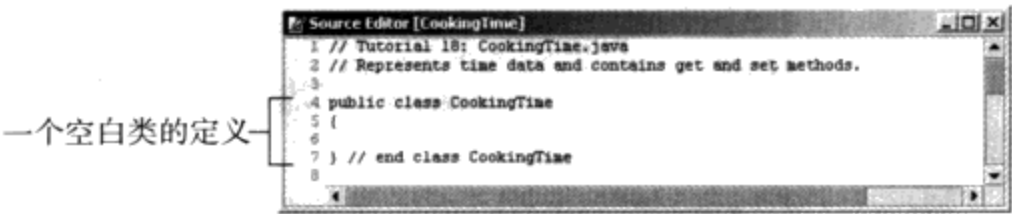


图 18.8 一个空白类的定义



好的编程习惯

在程序员定义的类的开始位置处添加注释,将有助于提高代码的可读性。注释的内容应说明包含这个类定义的文件名以及该类的作用。

4. 在应用程序中添加实例变量 将图 18.9 中第 6 行至第 8 行添加到 CookingTime 类中。第 7 行至第 8 行将声明两个 int 型实例变量——minute 和 second。在 CookingTime 类中存储了包含分钟和秒的时间值——分钟存储在 minute 中,秒存储在 second 中。回顾教程 14,实例变量的作用域为其所在类的整个区域。也就是说,在 CookingTime 类中声明的所有方法都能够访问到实例变量 minute 和 second。

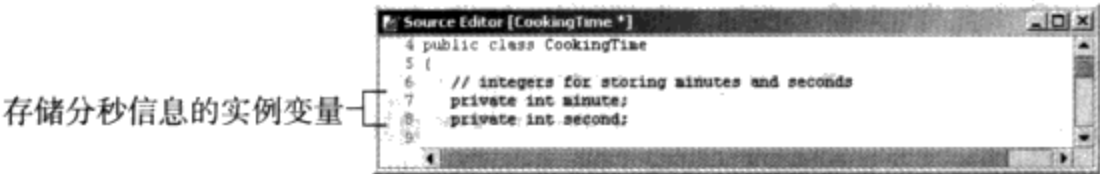


图 18.9 CookingTime 的实例变量

5. 保存应用程序 保存修改后的源代码文件。

自测题

1. 关键字 _____ 表示一个类定义的开始。
a) declare b) new c) class d) 以上答案均不对
2. 类的定义必须包含在一对 _____ 内。
a) 方括号 b) 花括号 c) 逗号 d) 圆括号

答案: 1) c 2) b

18.3 对象的初始化：构造方法

类中可包含方法和实例变量。试着回想一些曾经使用过的方法,诸如, DecimalFormat 类的 format 方法和 Random 类的 nextInt 方法,这些方法均属于各自所在的类。然而,构造方法则是一个用来初始化实例变量的方法,且构造方法同包含它的类同名。构造方法的定义同普通方法的定义类似,所不同的是构造方法没有返回值类型(也不允许使用 void)。在教程 15 中,曾学习过如何使用构造方法来创建对象。下面,将为 CookingTime 类定义一个构造方法,通过它创建 CookingTime 的对象并完成对对象实例变量的初始化工作。

定义一个构造方法

1. 在类中添加构造方法 将图 18.10 中第 10 行至第 14 行添加到 CookingTime 类的类体中。这些代码行为 CookingTime 类定义了一个构造方法。可以看到,该构造方法的方法名(CookingTime)与其所属类的类名相同。构造方法每当实例化(创建)类的一个对象时被调用一次。我们将按照以下步骤为构造方法添加需要的语句体。构造方法中的语句通常用于初始化实例变量。

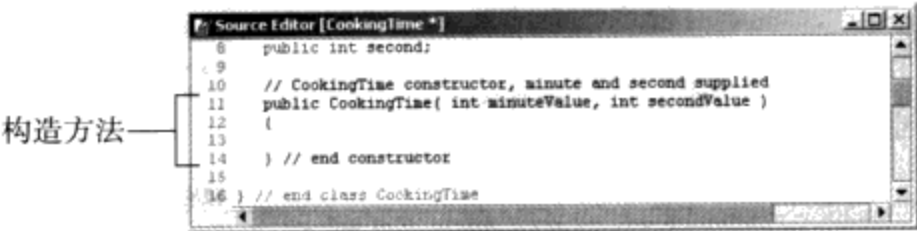


图 18.10 定义一个空的构造方法

图18.10中的这个构造方法,将接收两个参数。我们很快会看到如何为这一构造方法提供所需要的参数。构造方法返回一个类的对象(比如,构造方法 `CookingTime` 返回一个 `CookingTime` 类的实例)。构造方法不能指定返回值类型,这是构造方法同普通方法之间的重要区别。类的实例变量可通过其构造方法来初始化,或者是在定义类时被初始化。例如,变量 `second` 可在其声明处(参见第8行)被初始化,或者是在 `CookingTime` 构造方法的内部被初始化。

- 2. 在构造方法中初始化变量 将图 18.11 中第 13 行至第 14 行添加到构造方法中。这些代码行将把 `CookingTime` 的实例变量初始化为该构造方法的参数值(参见第 11 行)。当一个对象被创建时,通常

需要为对象指定某些值。例如,创建一个 `CookingTime` 对象(如 `microwaveTime`)可使用下面的代码:

`microwaveTime = new CookingTime(0,0);`
当这个 `CookingTime` 对象被创建时,其构造方法也将得到执行。通过将两个 0 传递给构造方法中的参数,该构造方法将利用它们完成了 `minute` 和 `second` 的初始化操作。

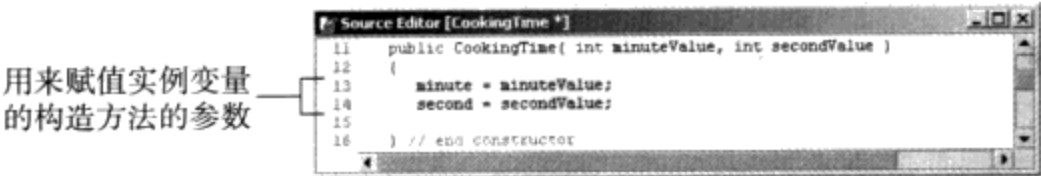


图 18.11 初始化实例变量的构造方法



错误预防提示

通过为构造方法提供有意义的值以初始化每一个对象将有助于减少逻辑错误的发生。

- 3. 保存应用程序 保存修改后的源代码文件。
- 4. 打开微波炉模拟应用程序的模板文件 在自己的文本编辑器中打开模板文件 `MicrowaveOven.java`。
- 5. 创建一个 `CookingTime` 对象 由于已完成 `CookingTime` 类的定义,现在,便可以声明一些属于 `CookingTime` 类型的对象。将图 18.12 中第 42 行至第 43 行添加到应用程序中。

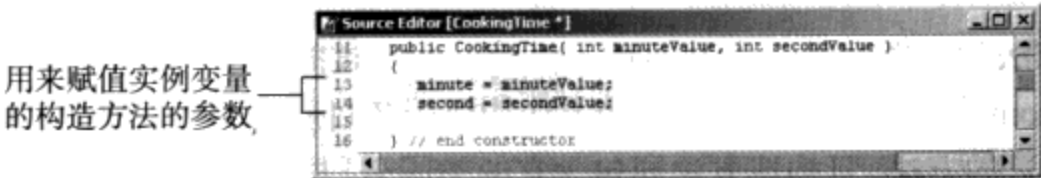


图 18.12 声明一个 `CookingTime` 类型的对象

注意,类名 `CookingTime` 是作为一种类型来使用的。正如可通过某个基本类型(如 `int`)创建多个变量一样,同样也可通过一个类(属于引用类型)创建多个对象。还可以按照需求创建属于自己的类,因为通过使用新的类便可完成对它的“扩充”,所以 Java 被认为是一种可以扩充的语言。
注意观察,实例变量 `microwaveTime` 的初始化和声明(参见第 43 行)是在同一行上来完成的。在关键字 `new` 之后,利用构造方法 `CookingTime` 便完成了这个 `CookingTime` 对象实例变量的初始化操作。通过将两个 0 传递 `CookingTime` 构造方法的参数中,便可把 `CookingTime` 的实例变量 `minute` 和 `second` 分别初始化成各自的 0 值。该表达式得到的是一个新创建的对象引用;之后,将引用赋予 `microwaveTime`。

- 6. 保存应用程序 保存修改后的源代码文件。

自测题

1. _____ 语言是一种能够通过新的类进行“扩展”的语言。
a) 数据 b) 可扩充的 c) 类型 d) 可扩展的
2. 实例变量的初始化 _____。
a) 是在它们声明时完成的 b) 将被设置为默认值
c) 是在一个构造方法中完成的 d) 以上答案都正确

答案: 1) b 2) d

18.4 get 方法和 set 方法

所有类的客户程序通常都希望能操作其实例变量。比如,假设某类中存储了一些与人有关的信息,其中包括年龄信息(存储在 `int` 型实例变量 `age` 中)。创建 `Person` 类对象的这个客户程序,很可能会修改这个 `age`——其结果就有可能是错误地将一个负值赋予给了该 `age`。一些类通过提供 `get` 和 `set` 方法(有时也称访问器和修改器),允许客户程序以一个较为常规和安全的方式对实例变量进行访问和修改。`get` 方法(访问器)通常用于检索对象中的某个值。`set` 方法(修改器)通常用于修改对象中的数据。在前面的教程中已经看到并使用过一些这样的方法,例如, `JLabel` 中作为检索或修改所显示文本的 `setText` 方法和 `getText` 方法。

`get` 方法允许客户程序取得某个实例变量的值。当执行代码:

```
minuteValue = timeObject.getMinute();
```

时, `getMinute` 方法将返回 `timeObject` 的实例变量 `minute` 的值,随后又将其赋予 `minuteValue`。

`set` 方法允许客户程序设置(即赋予)某个实例变量的值。例如,当代码:

```
timeObject.setMinute( 35 );
```

执行时, `setMinute` 方法会向实例变量 `minute` 赋予一个新值。一个 `set` 方法可能,或者说应该仔细去查看试图修改变量值的任何操作。其目的是确保所赋予的新值对于相应的数据项来说是合适的。利用这种方式来维护对象的数据,则认为数据是处在一个一致状态之中。方法 `setMinute` 通过将一个有效值赋予 `minute`,而使变量 `minute` 保存在一致状态之中(甚至是向 `set` 方法传递无效的数据)。用户只能在 0~59 的范围内为分钟指定某个值,所有不在该范围内的值都将被 `set` 方法丢弃并相应的为 `minute` 赋予 0 值。



好的编程习惯

尽管不是必需的,但在相应的 `get` 方法和 `set` 方法中保留单词 `get` 和 `set` 则是一个好的习惯。

在本教中,将学习如何创建自己的 `get` 方法和 `set` 方法,利用这些方法允许类的客户程序读取并修改其对象的实例变量。我们需为 `CookingTime` 类创建 4 个方法: `getMinute`, `setMinute`, `getSecond` 和 `setSecond`。

为 `CookingTime` 类定义方法

1. 定义 `minute` 的 `get` 方法 将图 18.12 中第 18 行至第 23 行添加到 `CookingTime` 类中。第 19 行起始于关键字 `public`。`public` 关键字是一个访问修饰符,它允许 `MicrowaveOven` 实例的客户程序调用其修饰的方法(`getMinute`)。在关键字 `public` 之后的返回值类型(`int`),表示该方法将返回一个 `int` 值,之后,可跟一个可带有参数列表的方法名(`getMinute`),对于 `get` 方法来说,其参数列表通常为`空`。当调用 `getMinute` 方法时,会返回 `minute` 的值,因此,在关键字 `return` 之后还应使用标识符 `minute` (参见第 21 行)。

从get方法中返回一个值

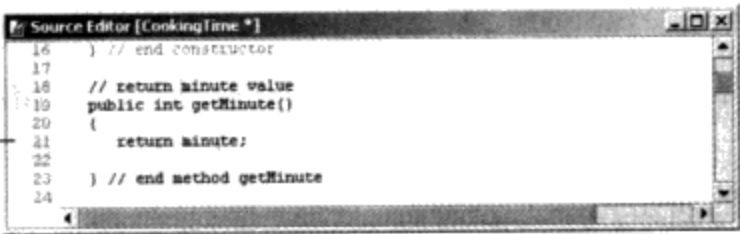


图 18.13 定义 getMinute 方法

2. 定义 minute 的 set 方法 将图 18.14 中第 25 行至第 37 行添加到 CookingTime 类中。此方法首先将 minute 值设置为其所接收到的 int 型参数，但还应测试该参数的值以确保其是否有效。方法 setMinute 所接受的值应小于 60 而大于 0，测试条件位于第 28 行。如果该参数 (value) 有效，将其赋予第 30 行上的 minute。否则，将 0 赋予第 34 行上的 minute。这会使 minute 的值保持一致状态之中。

可用来验证数据的代码

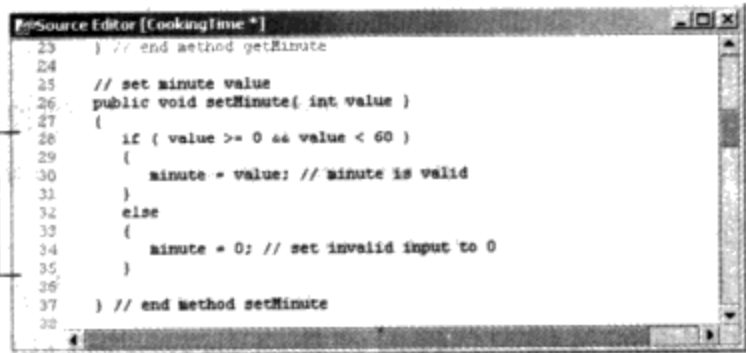


图 18.14 定义 setMinute 方法

3. 定义 second 的 get 方法 将图 18.15 中第 39 行至第 44 行添加到应用程序中。方法 getSecond 将返回 second 的值，其结果类似步骤 3 中 getMinute 方法所返回的 minute 值。

返回second值的代码

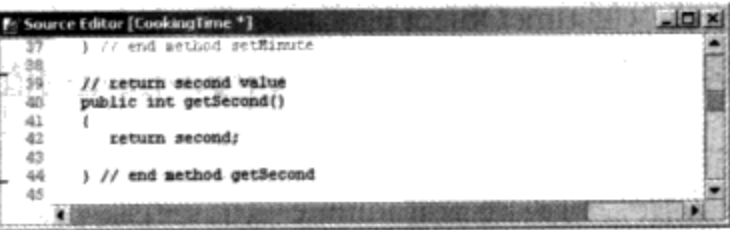


图 18.15 定义 getSecond 方法

4. 定义 second 的 set 方法 将图 18.16 中第 46 行至第 58 行添加到 CookingTime 类的定义中。可以看到，second 的 set 方法和 get 方法同 minute 的 set 方法和 get 方法类似，但与 minute 变量所不同的是，变量 second 将随时得到访问。

与setMinute方法相类似的setSecond方法

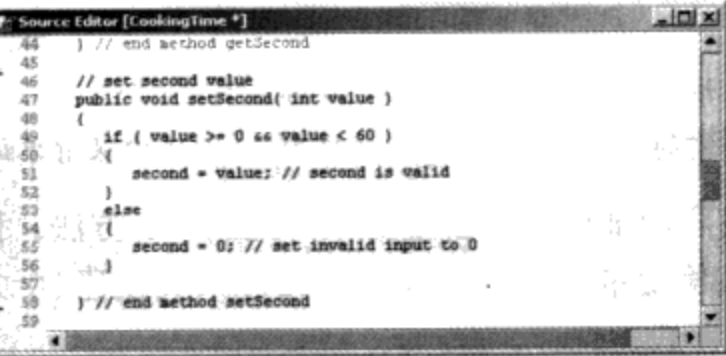


图 18.16 定义 setSecond 方法

5. 添加 isDone 方法 将图 18.17 中第 60 行至第 65 行添加到 CookingTime 类的定义中。方法 isDone 将返回一个 boolean 值。当时间到达 0 时，返回 true。否则，返回 false。第 63 行测试 minute 和 second 是否都等于 0。如果它们都等于 0，则返回 true。否则，返回 false。
6. 使用 set 方法 修改图 18.11 中第 13 行至第 14 行 (参见图 18.18)，把直接向 minute 和 second 中存储值的代码改为相应 set 方法的调用。既然已经定义 set 方法，就应在类的构造方法中使用这些 set 方法完成相应实例变量的初始化工作，从而确保只有有效值才可赋予 minute 和 second。

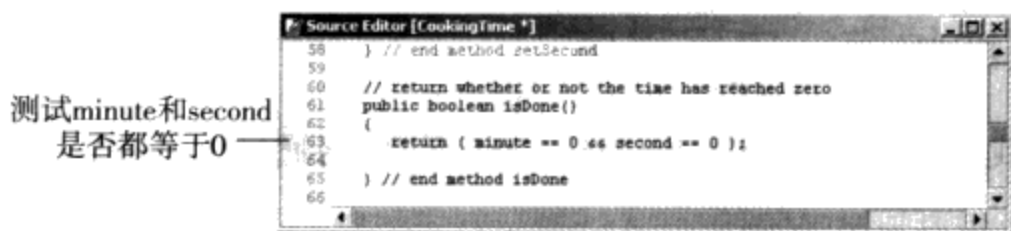


图 18.17 定义 isDone 方法

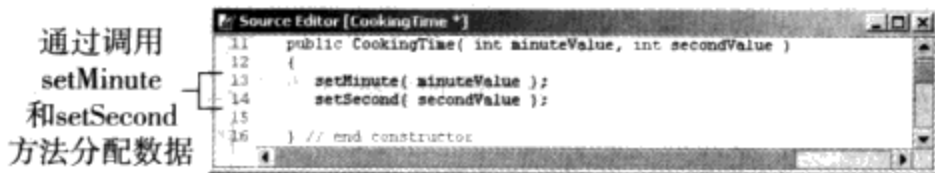


图 18.18 通用使用 set 方法来完成变量初始化的构造方法

调用自己编写的大多数方法，均是通过使用类名加一个点（如 `Integer.parseInt`）或者是变量名加一个点（如 `calculateJButton.setBounds`）的方式完成的。读者可能会注意到，某些方法的调用并不要求有一个类名或是对象名，如图 18.18 中第 13 行上针对 `setMinute` 方法的调用。在第 13 行上的方法调用过程中，隐含地调用了正被创建的某个对象的 `setMinute` 方法。在一个方法的内部，通过使用关键字 `this`，可显式地引用（也称 `this` 引用）某类之上一个进行调用的方法的对象。例如，第 13 行可以写为：

```
this.setMinute( minuteValue );
```

当编译程序碰到如第 13 行上的方法调用时，它会自动地在该方法调用的开始位置处添加 `"this."`。

7. 定义 tick 方法 将 18.19 图中第 67 行至第 82 行添加到 `CookingTime` 类的定义中。方法 `tick` 会每隔一秒对 `CookingTime` 的对象数据进行递减。`MicrowaveOven` 类将每隔一秒调用一次 `tick` 方法，从而模拟实际微波炉上的定时器。如果定时器的秒值大于 0（参见第 71 行），则将该数值减 1（参见第 73 行）。如果秒值为 0 而分钟值大于 0（参见第 76 行），则将分钟值减 1（参见第 78 行）并将秒值设置为 59（参见第 79 行）。
8. 保存应用程序 保存修改后的源代码文件。
9. 打开命令提示符窗口改变目录 选择 `Start → Programs → Accessories → Command Prompt` 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\MicrowaveOven` 进入到当前的工作目录中。
10. 编译该类 通过键入 `javac CookingTime.java` 对其进行编译。如果应用程序不能正确编译，在进入下一小节之前请先修改其中的错误代码。
11. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

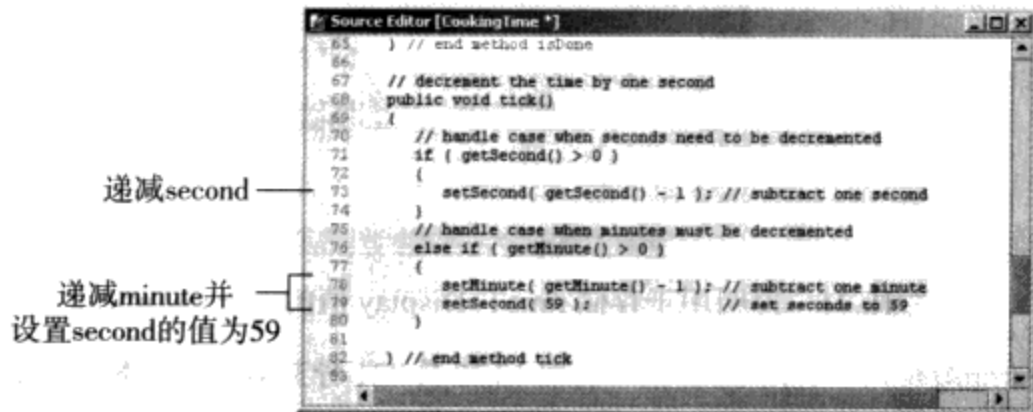


图 18.19 在 tick 方法中递减时间

自测题

1. _____ 可确保数据将以一致状态来保存。
a) get 方法 b) return 语句 c) boolean 值 d) set 方法
2. get 方法也称为 _____。
a) 构造方法 b) 访问修饰符 c) 访问器 d) 修改器

答案：1) d 2) c

18.5 完成微波炉模拟应用程序

到目前为止，我们已经完成了CookingTime类，下面，将通过使用该类的对象来保存所需的烹调时间。

完成微波炉模拟应用程序

- 1. 设定 Timer 的延迟 参照图 18.20 修改 MicrowaveOven.java 文件中第 326 行。Timer 类的对象会在指定时间间隔完成时产生一个 ActionEvent，这类似于时钟嘀嗒声的产生。两次嘀嗒声间隔的时间将作为 Timer 构造方法中的第一个参数，注意时间单位为毫秒。在应用程序中，Timer 将每隔 1000 毫秒（1 秒）生成一个事件。第二个参数（timerActionListener）已在模板中声明，用来处理由 clockTimer 产生的事件。

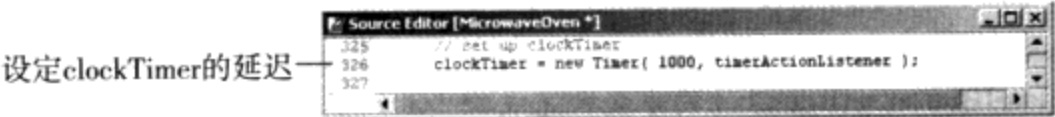


图 18.20 创建延迟时间为 1000 毫秒的 clockTimer

- 2. 格式化 currentTime 删除第 408 行上返回空字符串的那条 return 语句并将图 18.21 中第 408 行至第 424 行添加到 formatTime 中。第 409 行声明了一个 String 型的 currentTime，用来存储完成 timeToDisplay 格式化之后的值（此变量已在模板第 40 行上声明）。第 412 行至第 415 行上的 for 语句将在 currentTime 的左侧添加相应的 "0"，直至其长度为 4。第 418 行至第 422 行是当 currentTime 的长度超过 4 个字符时，将其缩短为 4 个字符长。第 424 行是将结果 currentTime 返回给调用方法。为了能将 currentTime 缩短为 4 个字符，需使用 String 类的 substring 方法。该方法返回原字符串中一个指定部分的字符串，但是它不会修改原来的字符串。第 421 行通过调用方法 substring 并为其传递两个参数，所得结果为 currentTime 中的前 4 个字符，最后再将这一结果赋值给 currentTime。第一个参数 0，表示该方法所返回字符串的起始位置为 currentTime 中第一个字符（其位置 0）。第二参数 4，表示所返回的这个字符串应结束于位置 4 之前的字符。在该行执行之后，currentTime 中将只包含 4 个字符。

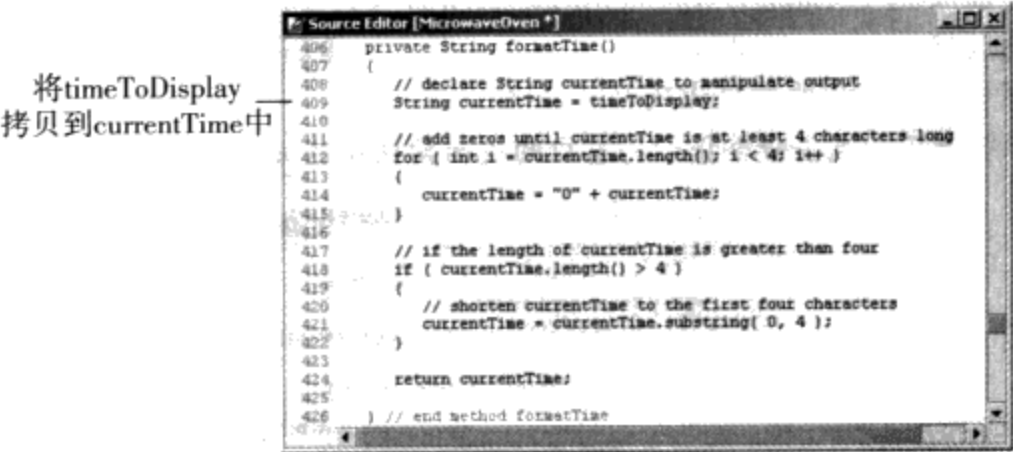


图 18.21 声明用于存储 timeToDisplay 值的 currentTime

- 3. 完成 startJButtonActionPerformed 方法 将图 18.22 第 431 行至第 445 行添加到方法 startJButtonActionPerformed 中。第 432 行通过调用 formatTime 方法，将返回的字符串存储在 fourDigitTime 中。第 435 行至第 436 行声明了两个变量 minute 和 second，并为这些变量赋予用户所输入的分钟数和秒数。第 439 行至第 440 行则通过调用 CookingTime 对象（microwaveTime）的 setMinute 方法和 setSecond 方法，设置应用程序初始时的烹调时间。第 443 行至第 445 行则是将格式化后的初始烹调时间显示为一个字符串，其内容包括：表示分钟的两位数字、一个冒号（:）以及表示秒的另外两位数字。比如，输入的烹调时间为 3 分 20 秒，则通过第 443 行至第 445 行可显示为 "03:20"。每一个数字都将通过 microwaveTime 的 get 方法以及 timeFormat 的 format 方法完成正确的格式化操作。timeFormat 对象是一个 DecimalFormat 类的实例，用来将某个数转换成一个至少为两位数字的字符串。

调用String类的两个
不同版本的substring

```
Source Editor [MicrowaveOven *]
429 private void startJButtonActionPerformed( ActionEvent event )
430 {
431     // get the time as four digits
432     String fourDigitTime = formatTime();
433
434     // extract minutes and seconds
435     String minute = fourDigitTime.substring( 0, 2 );
436     String second = fourDigitTime.substring( 2 );
437
438     // initialize CookingTime object to time entered by user
439     microwaveTime.setMinute( Integer.parseInt( minute ) );
440     microwaveTime.setSecond( Integer.parseInt( second ) );
441
442     // display formatted starting time in displayJTextField
443     displayJTextField.setText( timeFormat.format(
444         microwaveTime.getMinute() ) + ":" + timeFormat.format(
445         microwaveTime.getSecond() ) );
446
447 } // end method startJButtonActionPerformed
```

图 18.22 从 currentTime 中提取分钟和秒

第 435 行和第 430 行使用了两个不同版本的 substring 方法。其中，第 435 行使用的这个版本，我们已在步骤 3 中看到过。该方法将返回 fourDigits 中以位置 0 为起点（该方法的第一个参数）、位置 2（该方法的第二个参数）为终点的一个子字符串。

第 436 行使用的是 substring 方法的第二个版本。该版本只接收一个指示子字符串起始索引的参数。这里，通过调用该方法可返回 fourDigits 中以索引 2 为起始位置并直达结束字符的一个子字符串。

回顾教程 14 中曾经讲过，类中可包含两个同名但不同参数集的方法——这被称为方法的重载。方法 substring 便是方法重载的一个例子。

4. 启动烹调过程 将图 18.23 中第 447 行至第 450 行添加到应用程序中。第 447 行清除用户的输入（timeToDisplay），使用户能够重新输入一个新的烹调时间（在当前的烹调周期结束以后）。Timer 类的 start 方法会在每个时间间隔（参见图 18.20 中第 326 行）结束后生成 ActionEvent。第 449 行通过调用 start 方法来开启 clockTimer。clockTimerActionPerformed 方法（将很快实现）每隔 1 秒调用一次。第 450 行是将 windowJPanel 的 background 属性设置为黄色，从而模拟微波炉中打开的灯光。

开启定时器并打开“灯光”
以表明微波炉正在烹调

```
Source Editor [MicrowaveOven *]
443 microwaveTime.getSecond() );
444
445 timeToDisplay = ""; // clear timeToDisplay for future input
446
447 clockTimer.start(); // start timer
448
449 windowJPanel.setBackground( Color.YELLOW ); // turn "light" on
450
451
```

图 18.23 开启微波炉模拟应用程序的倒计时

5. 清除烹调时间 将图 18.24 中第 457 行至第 461 行添加到方法 clearJButtonActionPerformed 中。Timer 类的 stop 方法将用于终止 clockTimer，从而阻止进一步产生额外的 ActionEvent。第 458 行通过调用 clockTimer 的 stop 方法来终止倒计时功能。第 459 行是将 displayJTextField 的文本设置为 "Microwave Oven"。第 460 行是将 String 型 timeToDisplay 的内容清除为一个空的字符串。第 461 行是将 JPanel 的 background 属性设置为其最初时的灰色，从而模拟出微波炉内关闭的灯光。

关闭微波炉
并复位变量

```
Source Editor [MicrowaveOven *]
455 private void clearJButtonActionPerformed( ActionEvent event )
456 {
457     // stop Timer and reset variables to their initial settings
458     clockTimer.stop();
459     displayJTextField.setText( "Microwave Oven" );
460     timeToDisplay = "";
461     windowJPanel.setBackground( new Color( 204, 204, 204 ) );
462 }
```

图 18.24 清除微波炉的输入

6. 显示正在输入的数据 将图 18.25 中第 468 行至第 479 行添加到方法 displayTime 中。调用该方法需要有一个 String 型的参数 digit，代表用户每一次输入的烹调时间中的一位数字。第 469 行，是将 digit 追加至 String 型的时间ToDisplay 的结尾，从而存储用户的输入。在第 472 行，通过调用方法 formatTime，将 String 型的时间ToDisplay 转化为 4 位数字。第 475 行至第 476 行通过声明两个变量 minute 和 second，作为存储 fourDigitTime 中相应的字符串部分。第 479 行通过方法 setText，显示该应用程序的初始烹调时间。

自测题

- 1. _____ 类的实例变量能够在每次时间间隔结束时生成相应的事件。
a) Timer b) Interval c) Generator d) Clock
- 2. 表达式 `example.substring(3, 4)` 返回的字符为 _____。
a) 起始于位置 3 的 4 个连续字符 b) 起始于位置 3 终止于位置 4 之前的字符
c) 位置 3 和位置 4 上的字符 d) 位置 3 上的字符，并重复 4 次

答案: 1) a 2) b

18.6 控制成员的访问

关键字 `public` 和 `private` 被称为访问修饰符。在本教程前面的内容中，我们曾经利用访问修饰符 `private` 声明过一些实例变量，同时还使用了访问修饰符 `private` 或 `public` 声明了一些方法。被声明为 `public` 的类成员，可在所定义类的外部来使用。Java 类库中的许多类，都是通过提供一些 `public` 方法完成了开发人员同其对象之间的交互（如调用 `JTextField` 的 `setText` 方法可在该 `JTextField` 中显示文本）。而访问由修饰符 `private` 所声明的实例变量或方法，则使其只能应用于同一个类的其他方法。试图从某个类的外部来访问其 `private` 成员，则会出现一个编译错误。在 Java 类库中，也包含了一些旨在为开发人员提供一定功能的 `private` 变量和 `private` 方法，但开发人员并不能够直接在应用程序中使用它们。通常来说，实例变量应声明为 `private`，而方法则应声明为 `public`。下面，将带领读者查看这些用来修饰成员的修饰符。



常见编程错误

凡试图从某类的外部访问一个 `private` 成员，将出现一个语法错误。

控制成员的访问

- 1. 打开应用程序 如果文件 `CookingTime.java` 已经关闭，在自己的文本编辑器中将其再次打开。
- 2. 查看 `CookingTime` 的实例变量 查看图 18.29 中第 7 行至第 8 行。这段代码声明了两个 `private` 实例变量：`minute` 和 `second`，表示该变量只能被 `CookingTime` 类的成员所访问。类的 `private` 实例变量只能被该类的方法所访问。这便为开发人员提供了一种如何使使用者访问或修改这些实例变量的完全控制。

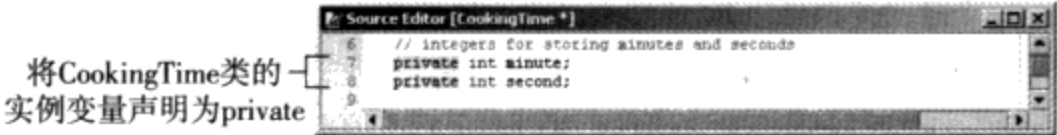


图 18.29 将 `CookingTime` 的实例变量声明为 `private`



软件设计提示

将类的所有实例变量都声明为 `private`。在必要时，通过提供由 `public` 修饰的 `get` 方法和 `set` 方法获取和设置其中的 `private` 实例变量。

- 3. 查看 `CookingTime` 的 `get` 方法和 `set` 方法 查看图 18.30 中，作为起始 `displayTime` 方法的第 466 行。同变量一样，凡声明为 `private` 的方法只能由同一个类的其他成员来访问。在这个例子中，因为只有 `MicrowaveOven` 类才允许使用该 `displayTime` 方法，因此应将该方法声明为 `private`。

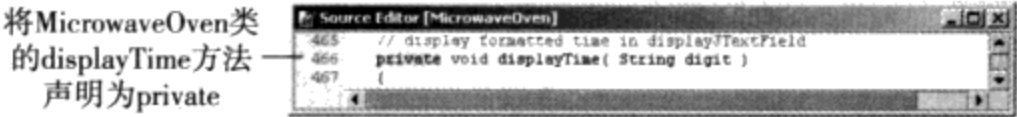


图 18.30 将 `displayTime` 方法声明为 `private`

- 4. 保存应用程序 保存修改后的源代码文件。

图 18.31 和图 18.32 中给出了微波炉模拟应用程序的完整源代码。在本教程中,凡需要添加、查看或者是修改的代码,均在图中相应的代码行中进行了突出显示。

```

1 // Tutorial 18: MicrowaveOven.java
2 // Mimics the behavior of a microwave oven.
3 import java.awt.*;
4 import java.awt.event.*;
5 import java.text.DecimalFormat;
6 import javax.swing.*;
7 import javax.swing.border.*;
8
9 public class MicrowaveOven extends JFrame
10 {
11     // JPanel for microwave window
12     private JPanel windowJPanel;
13
14     // JPanel for microwave controls
15     private JPanel controlJPanel;
16
17     // JTextField for cooking time
18     private JTextField displayJTextField;
19
20     // JButtons to set cooking time
21     private JButton oneJButton;
22     private JButton twoJButton;
23     private JButton threeJButton;
24     private JButton fourJButton;
25     private JButton fiveJButton;
26     private JButton sixJButton;
27     private JButton sevenJButton;
28     private JButton eightJButton;
29     private JButton nineJButton;
30     private JButton zeroJButton;
31
32     // JButtons to start and clear timer
33     private JButton startJButton;
34     private JButton clearJButton;
35
36     // Timer to count down seconds
37     private Timer clockTimer;
38
39     // String for storing digits entered by user
40     private String timeToDisplay = "";           声明 private 实例变量 timeToDisplay
41
42     // CookingTime instance for storing the current time
43     private CookingTime microwaveTime = new CookingTime( 0 , 0 );           声明 private 实例变量 cookingTime
44
45     // DecimalFormat to format time output
46     private DecimalFormat timeFormat = new DecimalFormat( "00" );
47
48     // no-argument constructor
49     public MicrowaveOven()
50     {
51         createUserInterface();
52     }
53
54     // create and position GUI components; register event handlers
55     private void createUserInterface()
56     {

```

```
57 // get content pane for attaching GUI components
58 Container contentPane = getContentPane();
59
60 // enable explicit positioning of GUI components
61 contentPane.setLayout( null );
62
63 // set up windowJPanel
64 windowJPanel = new JPanel();
65 windowJPanel.setBounds( 16 , 16 , 328 , 205 );
66 windowJPanel.setBorder( new LineBorder( Color.BLACK ) );
67 contentPane.add( windowJPanel );
68
69 // set up controlJPanel
70 controlJPanel = new JPanel();
71 controlJPanel.setBounds( 368 , 16 , 149, 205 );
72 controlJPanel.setBorder( new LineBorder( Color.BLACK ) );
73 controlJPanel.setLayout( null );
74 contentPane.add( controlJPanel );
75
76 // set up displayJTextField
77 displayJTextField = new JTextField();
78 displayJTextField.setBounds( 7 , 5 , 135 , 42 );
79 displayJTextField.setText( "Microwave Oven" );
80 displayJTextField.setHorizontalAlignment( JTextField.CENTER );
81 displayJTextField.setEditable( false );
82 controlJPanel.add( displayJTextField );
83
84 // set up oneJButton
85 oneJButton = new JButton();
86 oneJButton.setBounds( 13 , 59, 41, 24 );
87 oneJButton.setText( "1" );
88 controlJPanel.add( oneJButton );
89 oneJButton.addActionListener(
90
91     new ActionListener() // anonymous inner class
92     {
93         // event handler called when oneJButton is pressed
94         public void actionPerformed((ActionEvent event) )
95         {
96             oneJButtonActionPerformed( event );
97         }
98     } // end anonymous inner class
99 ); // end call to addActionListener
100
101 // set up twoJButton
102 twoJButton = new JButton();
103 twoJButton.setBounds( 54, 59, 41, 24 );
104 twoJButton.setText( "2" );
105 controlJPanel.add( twoJButton );
106 twoJButton.addActionListener(
107
108     new ActionListener() // anonymous inner class
109     {
110         // event handler called when twoJButton is pressed
111         public void actionPerformed((ActionEvent event) )
112         {
113             twoJButtonActionPerformed( event );
114         }
115     }
116 );
117
```



```
118         } // end anonymous inner class
119
120     ); // end call to addActionListener
121
122     // set up threeJButton
123     threeJButton = new JButton();
124     threeJButton.setBounds( 95, 59, 41, 24 );
125     threeJButton.setText( "3" );
126     controlJPanel.add( threeJButton );
127     threeJButton.addActionListener(
128
129         new ActionListener() // anonymous inner class
130     {
131         // event handler called when threeJButton is pressed
132         public void actionPerformed((ActionEvent event) )
133         {
134             threeJButtonActionPerformed( event );
135         }
136
137     } // end anonymous inner class
138
139 ); // end call to addActionListener
140
141 // set up fourJButton
142 fourJButton = new JButton();
143 fourJButton.setBounds( 13, 83, 41, 24 );
144 fourJButton.setText( "4" );
145 controlJPanel.add( fourJButton );
146 fourJButton.addActionListener(
147
148     new ActionListener() // anonymous inner class
149 {
150     // event handler called when fourJButton is pressed
151     public void actionPerformed((ActionEvent event) )
152     {
153         fourJButtonActionPerformed( event );
154     }
155
156 } // end anonymous inner class
157
158 ); // end call to addActionListener
159
160 // set up fiveJButton
161 fiveJButton = new JButton();
162 fiveJButton.setBounds( 54, 83, 41, 24 );
163 fiveJButton.setText( "5" );
164 controlJPanel.add( fiveJButton );
165 fiveJButton.addActionListener(
166
167     new ActionListener() // anonymous inner class
168 {
169     // event handler called when fiveJButton is pressed
170     public void actionPerformed((ActionEvent event) )
171     {
172         fiveJButtonActionPerformed( event );
173     }
174
175 } // end anonymous inner class
176
177 ); // end call to addActionListener
178
```

```
179 // set up sixJButton
180 sixJButton = new JButton();
181 sixJButton.setBounds( 95, 83 , 41, 24 );
182 sixJButton.setText( "6" );
183 controlJPanel.add( sixJButton );
184 sixJButton.addActionListener(
185
186     new ActionListener() // anonymous inner class
187     {
188         // event handler called when sixJButton is pressed
189         public void actionPerformed((ActionEvent event) )
190         {
191             sixJButtonActionPerformed( event );
192         }
193
194     } // end anonymous inner class
195
196 ); // end call to addActionListener
197
198 // set up sevenJButton
199 sevenJButton = new JButton();
200 sevenJButton.setBounds( 13 , 107, 41, 24 );
201 sevenJButton.setText( "7" );
202 controlJPanel.add( sevenJButton );
203 sevenJButton.addActionListener(
204
205     new ActionListener() // anonymous inner class
206     {
207         // event handler called when sevenJButton is pressed
208         public void actionPerformed((ActionEvent event) )
209         {
210             sevenJButtonActionPerformed( event );
211         }
212
213     } // end anonymous inner class
214
215 ); // end call to addActionListener
216
217 // set up eightJButton
218 eightJButton = new JButton();
219 eightJButton.setBounds( 54, 107, 41, 24 );
220 eightJButton.setText( "8" );
221 controlJPanel.add( eightJButton );
222 eightJButton.addActionListener(
223
224     new ActionListener() // anonymous inner class
225     {
226         // event handler called when eightJButton is pressed
227         public void actionPerformed((ActionEvent event) )
228         {
229             eightJButtonActionPerformed( event );
230         }
231
232     } // end anonymous inner class
233
234 ); // end call to addActionListener
235
236 // set up nineJButton
237 nineJButton = new JButton();
238 nineJButton.setBounds( 95, 107, 41, 24 );
239 nineJButton.setText( "9" );
```

```
240 controlJPanel.add( nineJButton );
241 nineJButton.addActionListener(
242     new ActionListener() // anonymous inner class
243     {
244         // event handler called when nineJButton is pressed
245         public void actionPerformed((ActionEvent event) )
246         {
247             nineJButtonActionPerformed( event );
248         }
249     } // end anonymous inner class
250 ); // end call to addActionListener
251
252 // set up zeroJButton
253 zeroJButton = new JButton();
254 zeroJButton.setBounds( 54, 131, 41, 24 );
255 zeroJButton.setText( "0" );
256 controlJPanel.add( zeroJButton );
257 zeroJButton.addActionListener(
258     new ActionListener() // anonymous inner class
259     {
260         // event handler called when zeroJButton is pressed
261         public void actionPerformed((ActionEvent event) )
262         {
263             zeroJButtonActionPerformed( event );
264         }
265     } // end anonymous inner class
266 ); // end call to addActionListener
267
268 // set up startJButton
269 startJButton = new JButton();
270 startJButton.setBounds( 7, 171, 64, 24 );
271 startJButton.setText( "Start" );
272 controlJPanel.add( startJButton );
273 startJButton.addActionListener(
274     new ActionListener() // anonymous inner class
275     {
276         // event handler called when startJButton is pressed
277         public void actionPerformed((ActionEvent event) )
278         {
279             startJButtonActionPerformed( event );
280         }
281     } // end anonymous inner class
282 ); // end call to addActionListener
283
284 // set up clearJButton
285 clearJButton = new JButton();
286 clearJButton.setBounds( 79, 171, 64, 24 );
287 clearJButton.setText( "Clear" );
288 controlJPanel.add( clearJButton );
289 clearJButton.addActionListener(
290     new ActionListener() // anonymous inner class
```

```
301     {
302         // event handler called when clearJButton is pressed
303         public void actionPerformed((ActionEvent event) )
304         {
305             clearJButtonActionPerformed( event );
306         }
307     } // end anonymous inner class
308 } // end call to addActionListener
309
310 // set up timerActionListener
311 ActionListener timerActionListener =
312     new ActionListener() // anonymous inner class
313     {
314         // event handler called every 1000 milliseconds
315         public void actionPerformed((ActionEvent event) )
316         {
317             clockTimerActionPerformed( event );
318         }
319     }; // end anonymous inner class
320
321 // set up clockTimer
322 clockTimer = new Timer( 1000, timerActionListener );
323
324 // set properties of application's window
325 setTitle( "Microwave Oven" ); // set title bar string
326 setSize( 536, 261 ); // set window size
327 setVisible( true ); // display window
328
329 } // end method createUserInterface
330
331 // add digit 1 to timeToDisplay
332 private void oneJButtonActionPerformed((ActionEvent event) )
333 {
334     displayTime( "1" ); // display time input properly
335 } // end method oneJButtonActionPerformed
336
337 // add digit 2 to timeToDisplay
338 private void twoJButtonActionPerformed((ActionEvent event) )
339 {
340     displayTime( "2" ); // display time input properly
341 } // end method twoJButtonActionPerformed
342
343 // add digit 3 to timeToDisplay
344 private void threeJButtonActionPerformed((ActionEvent event) )
345 {
346     displayTime( "3" ); // display time input properly
347 } // end method threeJButtonActionPerformed
348
349 // add digit 4 to timeToDisplay
350 private void fourJButtonActionPerformed((ActionEvent event) )
351 {
352     displayTime( "4" ); // display time input properly
353 } // end method fourJButtonActionPerformed
```



```

362
363 // add digit 5 to timeToDisplay
364 private void fiveJButtonActionPerformed((ActionEvent event) )
365 {
366     displayTime( "5" ); // display time input properly
367
368 } // end method fiveJButtonActionPerformed
369
370 // add digit 6 to timeToDisplay
371 private void sixJButtonActionPerformed((ActionEvent event) )
372 {
373     displayTime( "6" ); // display time input properly
374
375 } // end method sixJButtonActionPerformed
376
377 // add digit 7 to timeToDisplay
378 private void sevenJButtonActionPerformed((ActionEvent event) )
379 {
380     displayTime( "7" ); // display time input properly
381
382 } // end method sevenJButtonActionPerformed
383
384 // add digit 8 to timeToDisplay
385 private void eightJButtonActionPerformed((ActionEvent event) )
386 {
387     displayTime( "8" ); // display time input properly
388
389 } // end method eightJButtonActionPerformed
390
391 // add digit 9 to timeToDisplay
392 private void nineJButtonActionPerformed((ActionEvent event) )
393 {
394     displayTime( "9" ); // display time input properly
395
396 } // end method nineJButtonActionPerformed
397
398 // add digit 0 to timeToDisplay
399 private void zeroJButtonActionPerformed((ActionEvent event) )
400 {
401     displayTime( "0" ); // display time input properly
402
403 } // end method zeroJButtonActionPerformed
404
405 // format the time so that it has exactly four digits
406 private String formatTime()
407 {
408     // declare String currentTime to manipulate output
409     String currentTime = timeToDisplay;
410
411     // add zeros to currentTime until it is 4 characters long
412     for ( int i = currentTime.length(); i < 4 ; i++ )
413     {
414         currentTime = "0" + currentTime; // 必要时增加时间长度的显示
415     }
416
417     // if the length of currentTime is greater than four
418     if ( currentTime.length() > 4 )
419     {
420         // shorten currentTime to the first four characters

```

```

421         currentTime = currentTime.substring( 0 , 4 );    必要时缩短时间长度的显示
422     }
423
424     return currentTime;
425
426 } // end method formatTime
427
428 // start the microwave oven
429 private void startJButtonActionPerformed((ActionEvent event)
430 {
431     // get the time as four digits
432     String fourDigitTime = formatTime();    将时间以4位数字的形式进行存储
433
434     // extract minutes and seconds
435     String minute = fourDigitTime.substring( 0 , 2 )    取得分和秒
436     String second = fourDigitTime.substring( 2 );
437
438     // initialize CookingTime object to time entered by user
439     microwaveTime.setMinute( Integer.parseInt( minute ) );    为烹调时间
440     microwaveTime.setSecond( Integer.parseInt( second ) );    设置分和秒
441
442     // display formatted starting time in displayJTextField
443     displayJTextField.setText( timeFormat.format(    显示初
444         microwaveTime.getMinute() ) + ":" + timeFormat.format(    始时的
445         microwaveTime.getSecond() ) );    烹调时间
446
447     timeToDisplay = ""; // clear timeToDisplay for future input
448
449     clockTimer.start();    // start timer    启动微波炉
450     windowJPanel.setBackground( Color.YELLOW ); // turn "light" on
451
452 } // end method startJButtonActionPerformed
453
454 // clear the microwave oven
455 private void clearJButtonActionPerformed((ActionEvent event)
456 {
457     // stop Timer and reset variables to their initial settings
458     clockTimer.stop();
459     displayJTextField.setText( "Microwave Oven" );    终止微波炉并清除
460     timeToDisplay = "";    烹调时间的显示
461     windowJPanel.setBackground( new Color( 204 , 204 , 204 ) );
462
463 } // end method clearJButtonActionPerformed
464
465 // display formatted time in displayJTextField
466 private void displayTime( String digit )
467 {
468     // append digit to timeToDisplay
469     timeToDisplay += digit;    在时间中添加一位数字
470
471     // get the time as four digits
472     String fourDigitTime = formatTime();    将时间存储为4位数字
473
474     // extract minutes and seconds
475     String minute = fourDigitTime.substring( 0 , 2 );    取得分和秒
476     String second = fourDigitTime.substring( 2 );
477
478     // display number of minutes, ":", and then number of seconds

```

resetColor 方法 (已在模板第 605 行至第 613 行中提供给了读者), 把与所按键相关联的 JButton 的颜色设置为原来默认的灰色。

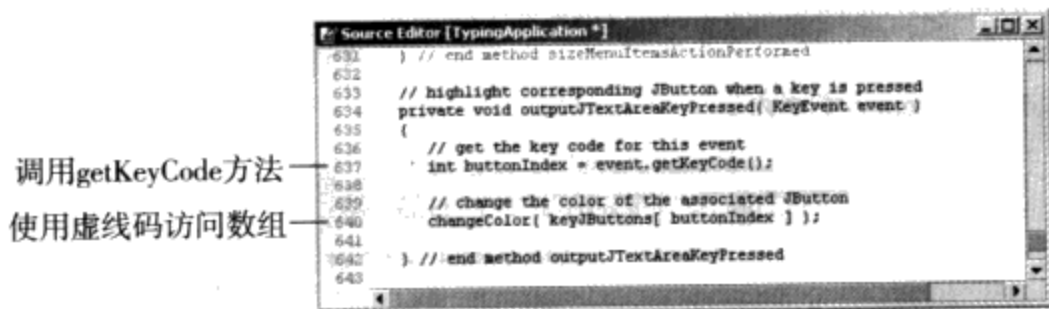


图 22.11 高亮显示与所按键相对应的 JButton

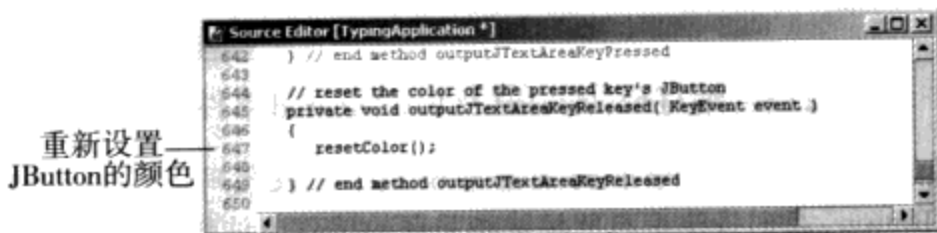


图 22.12 去除 JButton 上的高亮显示

9. 保存应用程序 保存修改后的源代码文件。

我们已经实现了高亮显示 JButton 并能在 JTextArea 中看到已输入的结果。下面, 将添加相应的代码, 允许用户对所显示结果的外观进行更改。通过在应用程序中添加一些 JMenu, 使用户执行如改变文本颜色、字号、字形等操作。

自测题

- 当按下或放开键盘上的一个按键时会产生 _____ 事件。
a) Keyboard b) KeyPressedEvent c) KeyChar d) KeyReleasedEvent
- 当释放一个按键时, 会调用 _____ 事件处理程序。
a) KeyEventReleased b) KeyUp c) KeyReleased d) 以上答案都不对

答案: 1) a 2) c

22.3 JMenu

利用菜单可以将应用程序中的一些相关命令组合在一起。尽管不同应用程序中的菜单及菜单命令都不尽相同, 但有一些菜单命令, 如打开命令和保存命令, 在大多数应用程序中都是一样的。菜单是 GUI 设计中的一个重要部分, 通过把多个命令组织在一起使 GUI 看起来不太“零乱”。但是, 菜单并非直接放置在某些组件的内部, 而通常是将它们添加到一个菜单条中, 菜单条一般位于应用程序界面的顶部。在这一节, 将学习如何添加 JMenu, 通过 JMenu 控制 JTextArea 内文本的颜色、字号及字体, 从而改进打字训练器应用程序。首先, 需要创建一个 JMenuBar, 然后再学习如何添加 JMenu 和 JMenuItem。

创建 Display 菜单

- 在应用程序中添加菜单条 将图 22.13 中第 93 行至第 95 行添加的程序代码, 创建一个 JMenuBar。JMenuBar 组件将作为应用程序菜单的容器。第 94 行创建了一个 JMenuBar(typingJMenuBar), 第 95 行通过调用继承自 JFrame 的 setJMenuBar 方法, 为应用程序设置其自身的 JMenuBar。

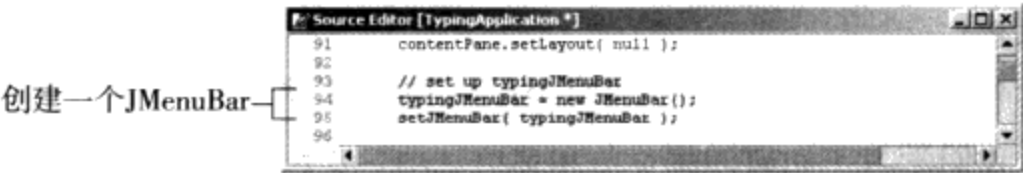


图 22.13 创建 JMenuBar

2. 向菜单条中添加菜单 将图 22.14 中第 97 行至第 100 行添加到程序代码中。第 98 行新创建了一个名为 displayJMenu 的 JMenu 组件，并在构造方法中传递一个将用于 JMenu 名称的字符串 "Display"。以前，都是通过调用 setText 方法来设置组件中所显示的文本。但在本应用程序中，我们通过向 JMenu 构造方法中传递一个字符串来设置显示的文本。这样做的目的是减少应用程序代码的数量。第 99 行通过调用 setMnemonic 方法并传递 KeyEvent 的常量 VK_D 将该组件的助记符设置为字符 D。教程 19 中曾经讲过，助记符允许用户使用键盘对某个组件行使选取操作。此时，若按下 Alt 键和 D 键，则等价于选择 Display JMenu。第 100 行使用 JMenuBar 的 add 方法将 displayJMenu 添加到 typingJMenuBar 中。所有菜单都是按照其添加顺序从左到右显示在菜单条中的。Display JMenu 位于应用程序 JMenuBar 的最左侧，凡直接添加在 JMenuBar 中的 JMenu 组件称为顶级菜单。

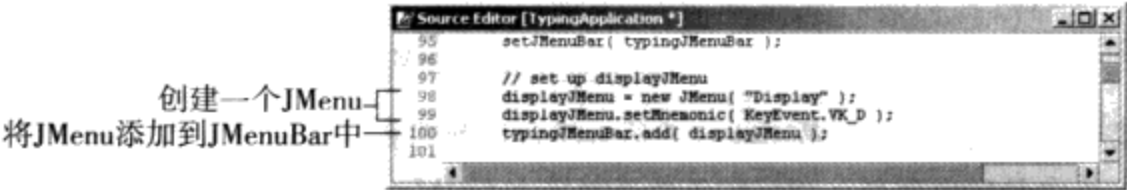


图 22.14 创建 displayJMenu

3. 创建一个菜单选项 将图 22.15 中第 104 行至第 106 行添加到程序代码中。第 103 行创建了一个新的 JMenuItem 组件，它是菜单内的第一个菜单选项。用户就是通过选择菜单选项来执行相应的操作。第 104 行为该 JMenuItem 设置助记符。第 105 行利用 JMenu 的 add 方法将 clearJMenuItem 添加到 displayJMenu 中。第 106 行则是使用 JMenu 类的 addSeparator 方法在 Clear Text 菜单选项的下面设置一个分隔条。

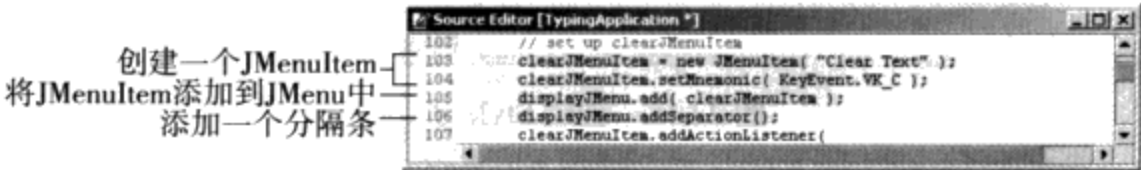


图 22.15 将一个 JMenuItem 添加到应用程序中

4. 添加 Color...菜单选项 将图 22.16 中第 123 行至第 124 行添加到程序代码中。第 122 行新创建了一个名为 colorJMenuItem 的 JMenuItem。第 123 行用来设置 colorJMenuItem 的助记符。第 124 行是将 colorJMenuItem 添加到 displayJMenu 中。此时，Color...菜单选项会出现在 Display JMenu 的底部，即位于分隔条的下方。

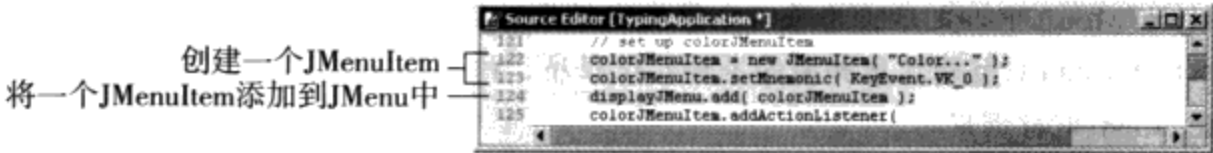


图 22.16 将 colorJMenuItem 添加到应用程序中

5. 保存应用程序 保存修改后的源代码文件。



GUI 设计提示

菜单选项中的文本应使用书名大写形式。



GUI 设计提示

利用分隔条分组管理 JMenu 中的 JMenuItem。



GUI 设计提示

如果点击某个菜单选项时会打开一个对话框，则在该菜单选项文本的后面添加一个省略号。

到目前为止，我们学习了如何创建菜单和菜单选项。接下来，将添加另外一个菜单并相应地创建一些其他菜单选项。

创建 Format 菜单

1. 创建第二个菜单 将图 22.17 中第 139 行至第 142 行添加到程序代码中。第 140 行创建了一个 Format JMenu 组件。第 141 行利用 KeyEvent 的常量 VK_F 设置此 JMenu 的助记符。第 142 行是将 formatJMenu 添加到 typingJMenuBar 中。因为 Format JMenu 是在添加完 Display JMenu 以后添加的，所以 Format JMenu 将位于 JMenuBar 中 Display JMenu 的右侧。

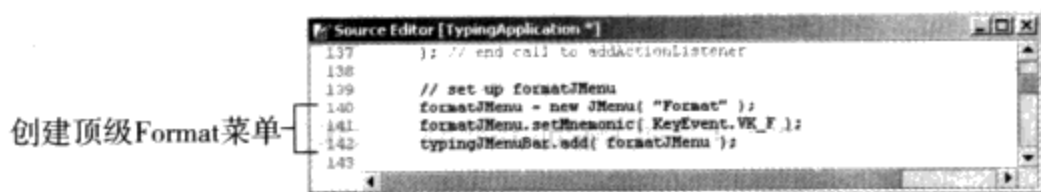


图 22.17 将第二个 JMenu 添加到应用程序中

2. 创建子菜单 将图 22.18 中第 144 行至第 147 行添加到程序代码中，创建一个名为 Style 的子菜单。子菜单也属于 JMenu 组件，但它们不能直接添加到 JMenuBar 中，而是添加在另外一个 JMenu 组件中。第 147 行将 styleJMenu 作为子菜单添加到 formatJMenu 中。注意，创建菜单和创建子菜单的语法是一样的，惟一区别是子菜单是添加在一个现有的 JMenu 中，而顶级菜单则是直接被添加到一个 JMenuBar 中。

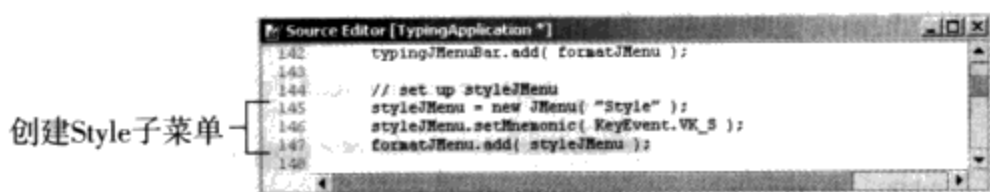


图 22.18 创建 Style 子菜单

3. 创建 JCheckBoxMenuItem 将图 22.19 中第 156 行添加到程序代码中。第 154 行至第 155 行新创建了一个 JCheckBoxMenuItem 组件。JCheckBoxMenuItem 组件的左侧有一个复选框，允许用户查看当前选项的选择情况。第 156 行将该 JCheckBoxMenuItem 添加到了一个 styleJMenu 中。styleMenuItems 和 styleNames 均为模板中已声明为实例变量的数组。变量 styleMenuItems 是 JCheckBoxMenuItem 类型的数组。变量 styleNames 则是存储字体名称（如 Bold 和 Italic）的 String 型数组。起始于第 152 行的 for 语句用于创建一些 JCheckBoxMenuItem，并将 styleMenuItems 数组中存储的每一个 JCheckBoxMenuItem 添加到 styleJMenu 中。之后，为每一个菜单选项注册其事件监听器。styleMenuItems 数组中的所有 JCheckBoxMenuItem 均使用相同的事件处理程序，该事件处理程序是在用户选取任何一个 JCheckBoxMenuItem 时被调用的。

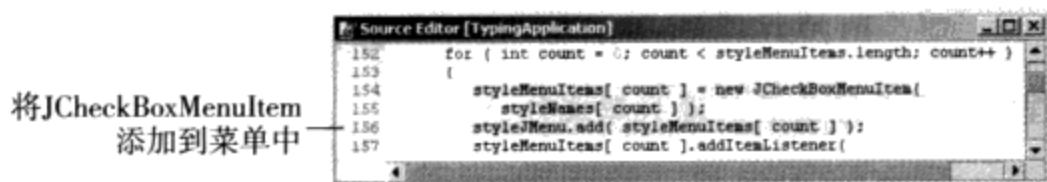


图 22.19 创建 JCheckBoxMenuItem



GUI 设计提示

JCheckBoxMenuItem 中的文本应具有描述性质且尽可能简短，同时还应使用书名大写形式。



GUI 设计提示

当用户需要从菜单中选择多个选项时，可以考虑使用 JCheckBoxMenuItem。

- 4. 创建子菜单 将图 22.20 中第 173 行至第 176 行添加到应用程序代码中，以创建一个 Size 子菜单。此段代码与读者创建 Style 子菜单所采用的代码相同。

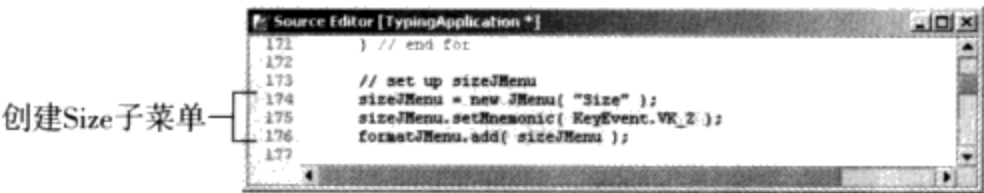


图 22.20 创建 Size 子菜单

- 5. 创建 JRadioButtonMenuItem 将图 22.21 中第 186 行至第 187 行添加到程序代码中。第 184 行至第 185 行新创建了一个 JRadioButtonMenuItem 组件。JRadioButtonMenuItem 组件的左侧有一个 JRadioButton，允许用户查看当前选项的选择情况。第 186 行将把每一个 JRadioButtonMenuItem 添加到 sizeJMenu 中。第 187 行通过在 sizeButtonGroup 中添加每一个 JRadioButtonMenuItem，确保任何时候只有一个 JRadioButtonMenuItem 能被选中。sizeMenuItems 和 sizeNames 均为模板中已作为实例变量声明的数组。变量 sizeMenuItems 是 JRadioButtonMenuItem 类型的数组。变量 sizeNames 则是存储不同字号（如 12，16 和 20）的 String 型数组。起始于第 182 行的 for 语句将创建一些 JRadioButtonMenuItem，并把 sizeMenuItems 数组中存储的每一个 JRadioButtonMenuItem 添加到 sizeJMenu 中。之后，为每一个菜单选项注册其事件监听器。sizeMenuItems 数组中的所有 JRadioButtonMenuItem 均使用相同的事件处理程序，该事件处理程序是在用户选择任何一个 JRadioButtonMenuItem 时被调用的。



GUI 设计提示

当用户需要从菜单中选择惟一的一个选项时，可以考虑使用 JRadioButtonMenuItem。

- 6. 保存应用程序 保存修改后的源代码文件。

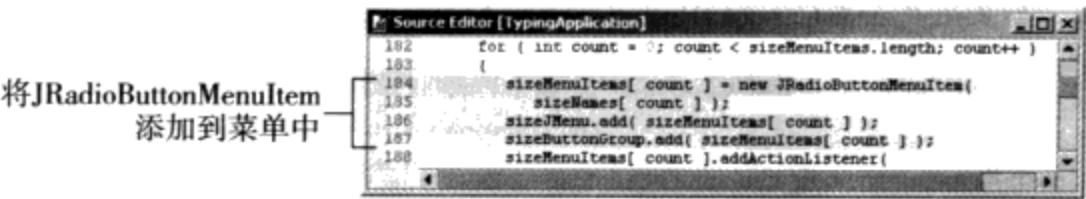


图 22.21 创建 Size JRadioButtonMenuItem

- 7. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\TypingApplication 进入到当前工作目录中。
- 8. 编译应用程序 键入 javac TypingApplication.java 编译该应用程序。
- 9. 运行应用程序 若能正确编译应用程序，键入 java TypingApplication 来运行它。此时，Display JMenu 和 Format JMenu 将出现在 JMenuBar 中（如图 22.22 所示）。
- 10. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- 11. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

自测题

- 1. JMenu 中包含了 ____。
a) 供用户选择的命令 b) 子菜单 c) 分隔条 d) 以上答案都正确
- 2. 子菜单是指添加在一个 ____ 组件中的 JMenu。
a) JMenu b) JMenuBar c) JMenuItem d) JSeparator

答案：1) d 2) a

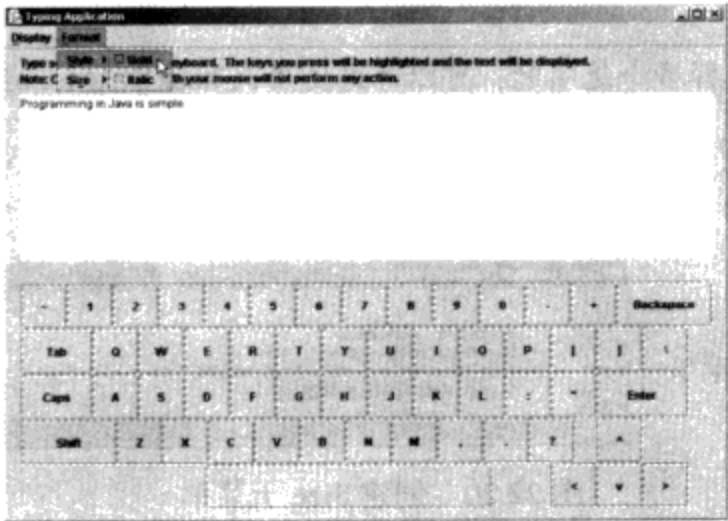


图 22.22 添有 JMenu 的打字训练器应用程序

22.4 JColorChooser

为了能在选择菜单选项时执行某些操作，必须为其添加事件处理程序。打字训练器应用程序中提供的菜单，允许用户对 JTextArea 中输出的文本指定颜色。下面，将学习如何使用 JColorChooser 选择颜色。JColorChooser 允许用户采用交互方式从颜色对话框中选择任何一个 Color。

编写 colorJMenuItemActionPerformed 方法

- 1. 打开 JColorChooser 将图 22.23 中第 650 行至第 651 行添加到 colorJMenuItemActionPerformed 方法中。此段代码通过调用 JColorChooser 类的 showDialog 方法打开一个允许用户从中选择颜色的对话框，相应的返回值也将赋值给 foregroundColor。传递至 showDialog 方法的首个参数为当前组件的父组件。当对话框出现时，可将其居于父组件之上。使用 this 引用表示这一 JColorChooser 对话框将放置在该应用程序 GUI 的上方。第二个参数表示显示在 JColorChooser 标题栏内的文本。第三个参数指定打开 JColorChooser 时的初始颜色。如果用户点击 OK JButton，showDialog 方法会返回一个表示所选显颜色的 Color 对象。如果用户点击 Cancel JButton，showDialog 方法则返回 null。

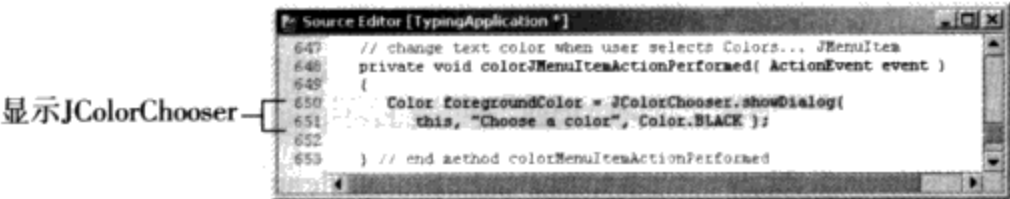


图 22.23 显示 JColorChooser 对话框

- 2. 设置文本颜色 将图 22.24 中第 653 行至第 658 行添加到程序代码中。第 654 行的 if 语句用于测试 foregroundColor 是否等于 null（当用户点击了 Cancel JButton 时）。第 657 行利用 setForeground 方法将文本颜色设置为用户通过 JColorChooser 选择的颜色。

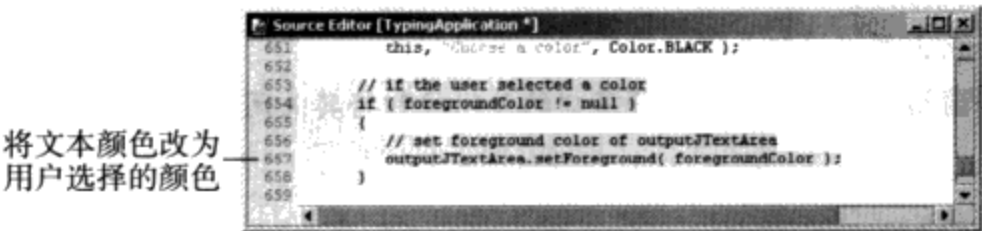


图 22.24 设置 JTextArea 中文本的颜色

- 3. 保存应用程序 保存修改后的源代码文件。

以下，实现 outputJTextArea 中文本字体的改变。在 Java 中，指定字体必须给出该字体的名称、字号和字体。本应用程序中，允许用户更改字号和字体。

改变字体

- 1. 声明存储字体的变量 将图 22.25 中第 665 行添加到程序代码中。该代码行将把变量 style 初始化为常量 Font.PLAIN。Font 类允许开发人员创建并管理代表字体的对象，同时它还提供了一些字体常量。常量 Font.PLAIN 代表常规字体（正常字体），即非斜体也非粗体。

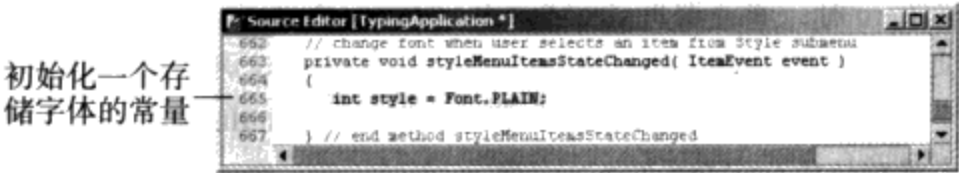


图 22.25 声明存储 Font 字体的变量

- 2. 修改字体 将图 22.26 中第 667 行至第 677 行添加到程序代码中。第 668 行检查 Bold JCheckBoxMenuItem 是否被选取。Bold JCheckBoxMenuItem 将作为 styleMenuItems 数组中的首个元素（其索引为 0）。如果用户选择了粗体，第 670 行会将常量 Font.BOLD 添加到变量 style 中（常量 Font.BOLD 代表粗体）。第 674 行检查 Italic JCheckBoxMenuItem 是否被选取。同时，Italic JCheckBoxMenuItem 将作为 styleMenuItems 数组中的第 2 个元素（索引为 1）。如果用户选择了斜体，第 676 行会将常量 Font.ITALIC 添加到变量 style 中（常量 Font.ITALIC 代表斜体）。

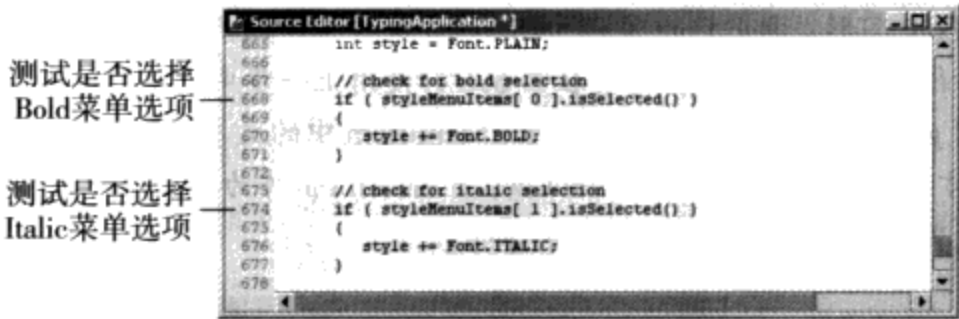


图 22.26 确定字体

- 3. 创建一个新的 Font 对象 添加图 22.27 中第 679 行至第 681 行，新创建一个 Font 类的对象。Font 构造方法中有 3 个参数，分别为：字体名称、字形和字号。这里，因为只需要改变字体，所以可将字体名称和字号保存下来。为保存字体名称，把 outputJTextArea 中的 Font 名称传递给构造方法。此 Font 已存储在实例变量 outputFont 之中（于模板第 70 行中声明）。读者可通过调用 Font 类的 getName 方法取得该 Font 的名称。getName 方法的返回值将作为第一个参数传递给该构造方法（参见第 680 行）。第二参数为 style，这样，用户对字体的改动将直接反应给新的 Font。另外，此处希望能够保存字号。利用 Font 类的 getSize 方法返回 Font 的字号，其结果将作为第三个参数传递给构造方法（参见第 681 行）。
- 4. 设置 outputJTextArea 中的字体 将图 22.28 中第 683 行添加到程序代码中。此代码行使用 setFont 方法设置 outputJTextArea 中的字体，并为该方法传递上一步中所创建的 Font 对象。

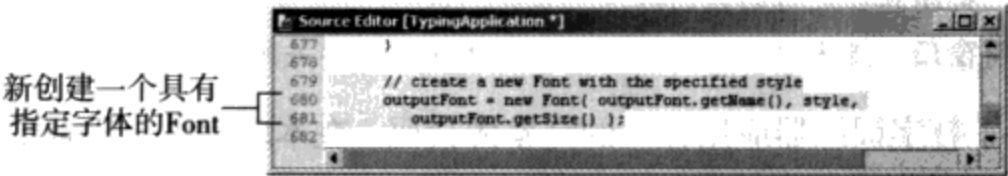


图 22.27 新创建一个具有指定字体的 Font

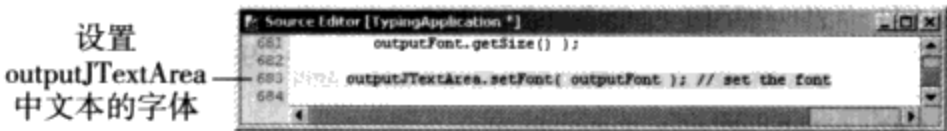


图 22.28 使用 setFont 方法改变文本字体

- 5. 保存应用程序 保存修改后的源代码文件。

下面将要添加的这个功能允许用户更改 JTextArea 中文本的字号。

更改字号

1. **查找事件源** 将图 22.29 中第 690 行至第 691 行添加到程序代码中。记得在应用程序的探试过程中，菜单 Format → Size 中有 3 个 JRadioButtonMenuItem，其值分别为 12，16 和 20。用户点击任意一个 JRadioButtonMenuItem 时，都将调用 sizeMenuItemActionPerformed 方法。可以利用 ActionEvent 类的 getSource 方法确定所选中的 JRadioButtonMenuItem。第 691 行通过调用 getSource 方法，返回一个 Object 类型的引用（事件源）。之后，需要将其造型为 JRadioButtonMenuItem 类型。用户所选择的这个 JRadioButtonMenuItem 会被赋值给 sizeMenuItem。

存储一个 JRadioButtonMenuItem 引用

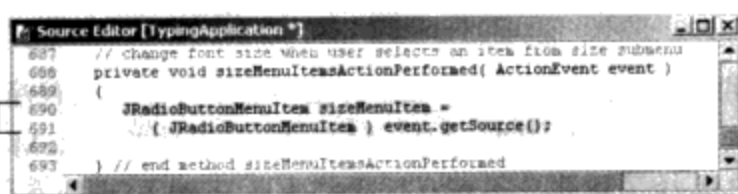


图 22.29 取得事件源的引用

2. **创建一个新的 Font 对象** 添加图 22.30 中第 693 行至第 698 行，创建一个新的 Font。此方法只需改变字号大小，因而还应保存字体名称和字形。通过 getName 方法为构造方法传递字体名称（参见第 694 行）。为了保存 Font 字形，调用 Font 类的 getStyle 方法并将返回值作为第二个参数传递给 Font 构造方法（参见第 695 行）。为确定字号，需要在第 696 行使用上一步中所创建的引用（sizeMenuItem）。此代码行通过调用 getText 方法，将 JRadioButtonMenuItem 中的文本以字符串形式返回。之后，再利用 Integer.parseInt 方法将字符串转换为一个 int 值，并将该 int 值作为第三个参数传递给 Font 的构造方法。第 698 行用于设置 outputJTextArea 中文本的字体。

创建一个具有指定字号的新 Font

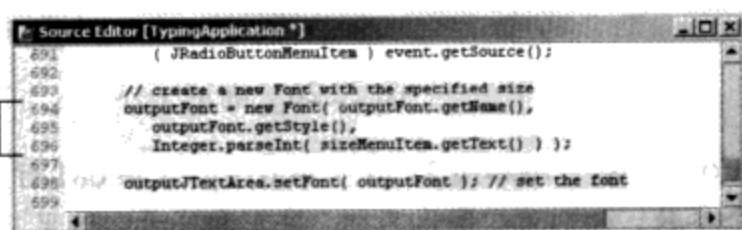


图 22.30 设置 Font 的新字号

3. **保存应用程序** 保存修改后的源代码文件。
4. **打开命令提示符窗口改变目录** 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\TypingApplication 进入到当前工作目录中。
5. **编译应用程序** 键入 javac TypingApplication.java 编译该应用程序。
6. **运行应用程序** 若能正确编译应用程序，键入 java TypingApplication 来运行它。
7. **点击关闭按钮** 关闭正在运行的应用程序
8. **关闭命令提示符窗口** 点击命令提示符窗口的关闭按钮将其关闭。

图 22.31 中给出了打字训练器应用程序的完整源代码。在本教程中，凡需要添加、查看或者是修改的代码，在图中相应的代码行中均进行了突出显示。

```
1 // Tutorial 22: TypingApplication.java
2 // Application enables users to practice typing
3 import java.awt.*;
4 import java.awt.event.*;
5 import javax.swing.*;
6 import java.text.*;
7
```

```

8 public class TypingApplication extends JFrame
9 {
10     // JMenuBar for display and format options
11     private JMenuBar typingJMenuBar;
12
13     // JMenu to show display options clear, invert colors and color
14     private JMenu displayJMenu;
15
16     // JMenuItem to clear the JTextArea and choose color
17     private JMenuItem clearJMenuItem;
18     private JMenuItem colorJMenuItem;
19
20     // JMenu to display format options style and size
21     private JMenu formatJMenu;
22
23     // JMenu and array of JCheckBoxMenuItems to display style options
24     private JMenu styleJMenu;
25     private JCheckBoxMenuItem styleMenuItems[];
26
27     // JMenu, array of JRadioButtonMenuItems and ButtonGroup to
28     // display size options
29     private JMenu sizeJMenu;
30     private JRadioButtonMenuItem sizeMenuItems[];
31     private ButtonGroup sizeButtonGroup;
32
33     // JLabel and JTextArea to display text output
34     private JLabel prompt1JLabel, prompt2JLabel;
35     private JTextArea outputJTextArea;
36
37     // JButtons to represent first row of keys
38     private JButton tildeJButton, oneJButton, twoJButton,
39         threeJButton, fourJButton, fiveJButton, sixJButton,
40         sevenJButton, eightJButton, nineJButton, zeroJButton,
41         hyphenJButton, plusJButton, backspaceJButton;
42
43     // JButtons to represent second row of keys
44     private JButton tabJButton, qJButton, wJButton, eJButton,
45         rJButton, tJButton, yJButton, uJButton, iJButton, oJButton,
46         pJButton, leftBraceJButton, rightBraceJButton, slashJButton;
47
48     // JButtons to represent third row of keys
49     private JButton capsJButton, aJButton, sJButton, dJButton,
50         fJButton, gJButton, hJButton, jJButton, kJButton, lJButton,
51         colonJButton, quoteJButton, enterJButton;
52
53     // JButtons to represent fourth row of keys
54     private JButton shiftLeftJButton, zJButton, xJButton, cJButton,
55         vJButton, bJButton, nJButton, mJButton, commaJButton,
56         periodJButton, questionJButton, upJButton;
57
58     // JButtons to represent fifth row of keys
59     private JButton spaceJButton, leftJButton, downJButton,
60         rightJButton;
61
62     // JButton to store the last JButton typed
63     private JButton lastJButton;
64
65     // array of JButtons
66     private JButton[] keyJButtons =
67         new JButton[ KeyEvent.KEY_LAST + 1 ];
68

```

以虚键码常量为索引的 JButton 数组

```

69 // Font of outputJTextArea
70 private Font outputFont;
71
72 // String array of font sizes
73 String sizeNames[] = { "12" , "16" , "20" };
74
75 // String array of font styles
76 String styleNames[] = { "Bold", "Italic" };
77
78 // no-argument constructor
79 public TypingApplication()
80 {
81     createUserInterface();
82 }
83
84 // set and position GUI components; register event handlers
85 private void createUserInterface()
86 {
87     // get content pane for attaching GUI components
88     Container contentPane = getContentPane();
89
90     // enable explicit positioning of GUI components
91     contentPane.setLayout( null );
92
93     // set up typingJMenuBar
94     typingJMenuBar = new JMenuBar(); // 创建 JMenuBar
95     setJMenuBar( typingJMenuBar );
96
97     // set up displayJMenu
98     displayJMenu = new JMenu( "Display" ); // 创建 JMenu
99     displayJMenu.setMnemonic( KeyEvent.VK_D );
100    typingJMenuBar.add( displayJMenu ); // 将 JMenu 添加到 JMenuBar 中
101
102    // set up clearJMenuItem
103    clearJMenuItem = new JMenuItem( "Clear Text" ); // 创建 JMenuItem
104    clearJMenuItem.setMnemonic( KeyEvent.VK_C );
105    displayJMenu.add( clearJMenuItem ); // 将 JMenuItem 添加到 JMenu 中
106    displayJMenu.addSeparator(); // 在 JMenu 中添加分隔条
107    clearJMenuItem.addActionListener(
108
109        new ActionListener() // anonymous inner class
110        {
111            // event handler called when clearJMenuItem is selected
112            public void actionPerformed((ActionEvent event) )
113            {
114                clearJMenuItemActionPerformed( event );
115            }
116
117        } // end anonymous inner class
118
119    ); // end call to addActionListener
120
121    // set up colorJMenuItem
122    colorJMenuItem = new JMenuItem( "Color..." ); // 创建另一个 JMenuItem
123    colorJMenuItem.setMnemonic( KeyEvent.VK_O );
124    displayJMenu.add( colorJMenuItem ); // 将 JMenuItem 添加到 JMenu 中
125    colorJMenuItem.addActionListener(
126
127        new ActionListener() // anonymous inner class
128        {

```

```

129         // event handler called when colorJMenuItem is selected
130         public void actionPerformed((ActionEvent event) )
131         {
132             colorJMenuItemActionPerformed( event );
133         }
134     } // end anonymous inner class
135 } // end call to addActionListener
136
137 // set up formatJMenu
138 formatJMenu = new JMenu( "Format" );
139 formatJMenu.setMnemonic( KeyEvent.VK_F );
140 typingJMenuBar.add( formatJMenu );
141                                     创建顶级 Format 菜单
142
143 // set up styleJMenu
144 styleJMenu = new JMenu( "Style" );
145 styleJMenu.setMnemonic( KeyEvent.VK_S );
146 formatJMenu.add( styleJMenu );
147                                     创建 Style 子菜单
148
149 styleMenuItems = new JCheckBoxMenuItem[ styleNames.length ];
150
151 // set up styleMenuItems
152 for ( int count = 0 ; count < styleMenuItems.length; count++ )
153 {
154     styleMenuItems[ count ] = new JCheckBoxMenuItem(
155         styleNames[ count ] );
156     styleJMenu.add( styleMenuItems[ count ] );
157     styleMenuItems[ count ].addItemListener(
158         new ItemListener() // anonymous inner class
159         {
160             // event handler called when styleMenuItems selected
161             public void itemStateChanged( ItemEvent event )
162             {
163                 styleMenuItemsStateChanged( event );
164             }
165         } // end anonymous inner class
166     ); // end call to addItemListener
167 } // end for
168
169 // set up sizeJMenu
170 sizeJMenu = new JMenu( "Size" );
171 sizeJMenu.setMnemonic( KeyEvent.VK_Z );
172 formatJMenu.add( sizeJMenu );
173                                     创建 Size 子菜单
174
175 sizeMenuItems = new JRadioButtonMenuItem[ sizeNames.length ];
176 sizeButtonGroup = new ButtonGroup();
177
178 // set up sizeMenuItems
179 for ( int count = 0 ; count < sizeMenuItems.length; count++ )
180 {
181     sizeMenuItems[ count ] = new JRadioButtonMenuItem(
182         sizeNames[ count ] );
183     sizeJMenu.add( sizeMenuItems[ count ] );
184     sizeButtonGroup.add( sizeMenuItems[ count ] );
185     sizeMenuItems[ count ].addActionListener(
186                                     创建 JRadioButtonMenuItem
187                                     将 JRadioButtonMenuItem
188                                     添加到菜单中

```

```

189         new ActionListener() // anonymous inner class
190     {
191         // event handler called when sizeMenuItems is selected
192         public void actionPerformed( ActionEvent event )
193         {
194             sizeMenuItemsActionPerformed( event );
195         }
196     } // end anonymous inner class
197
198     ); // end call to addActionListener
199
200 } // end for
201
202 // set up prompt1JLabel
203 prompt1JLabel = new JLabel( "Type some text using your " +
204     "keyboard. The keys you press will be highlighted and " +
205     "the text will be displayed." );
206 prompt1JLabel.setBounds( 15, 5, 725, 20 );
207 contentPane.add( prompt1JLabel );
208
209 // set up prompt2JLabel
210 prompt2JLabel = new JLabel( "Note: Clicking the buttons " +
211     "with your mouse will not perform any action." );
212 prompt2JLabel.setBounds( 15, 20, 725, 25 );
213 contentPane.add( prompt2JLabel );
214
215 // set up outputJTextArea
216 outputJTextArea = new JTextArea();
217 outputJTextArea.setBounds( 15, 50, 725, 175 );
218 outputJTextArea.setLineWrap( true );
219 contentPane.add( outputJTextArea );
220 outputFont = outputJTextArea.getFont();
221
222 outputJTextArea.addKeyListener( // 通过调用addKeyListener方法注册KeyListener对象
223     new KeyListener() // anonymous inner class
224     {
225         // event handler called when any key is pressed
226         public void keyPressed( KeyEvent event ) // 定义keyPressed事件处理程序
227         {
228             outputJTextAreaKeyPressed( event );
229         }
230
231         // event handler called when any key is released
232         public void keyReleased( KeyEvent event ) // 定义keyReleased事件处理程序
233         {
234             outputJTextAreaKeyReleased( event );
235         }
236
237         // event handler called when any key is typed
238         public void keyTyped( KeyEvent event ) // 定义空的keyTyped事件处理程序
239         {
240         }
241     } // end anonymous inner class
242
243 ); // end call to addKeyListener
244
245
246
247

```



```
248 outputJTextArea.addFocusListener(
249
250     new FocusAdapter() // anonymous inner class
251     {
252         // event handler called when outputJTextArea loses focus
253         public void focusLost( FocusEvent event )
254         {
255             outputJTextAreaFocusLost( event );
256         }
257     } // end anonymous inner class
258 ); // end call to addFocusListener
259
260 // set up tildeJButton
261 tildeJButton = new JButton( "~" );
262 tildeJButton.setBounds( 15, 250 , 48, 48 );
263 contentPane.add( tildeJButton );
264 keyJButtons[ KeyEvent.VK_BACK_QUOTE ] = tildeJButton;
265
266 // set up oneJButton
267 oneJButton = new JButton( "1" );
268 oneJButton.setBounds( 63, 250 , 48, 48 );
269 contentPane.add( oneJButton );
270 keyJButtons[ KeyEvent.VK_1 ] = oneJButton;
271
272 // set up twoJButton
273 twoJButton = new JButton( "2" );
274 twoJButton.setBounds( 111, 250 , 48, 48 );
275 contentPane.add( twoJButton );
276 keyJButtons[ KeyEvent.VK_2 ] = twoJButton;
277
278 // set up threeJButton
279 threeJButton = new JButton( "3" );
280 threeJButton.setBounds( 159, 250 , 48, 48 );
281 contentPane.add( threeJButton );
282 keyJButtons[ KeyEvent.VK_3 ] = threeJButton;
283
284 // set up fourJButton
285 fourJButton = new JButton( "4" );
286 fourJButton.setBounds( 207, 250 , 48, 48 );
287 contentPane.add( fourJButton );
288 keyJButtons[ KeyEvent.VK_4 ] = fourJButton;
289
290 // set up fiveJButton
291 fiveJButton = new JButton( "5" );
292 fiveJButton.setBounds( 255, 250 , 48, 48 );
293 contentPane.add( fiveJButton );
294 keyJButtons[ KeyEvent.VK_5 ] = fiveJButton;
295
296 // set up sixJButton
297 sixJButton = new JButton( "6" );
298 sixJButton.setBounds( 303, 250 , 48, 48 );
299 contentPane.add( sixJButton );
300 keyJButtons[ KeyEvent.VK_6 ] = sixJButton;
301
302 // set up sevenJButton
303 sevenJButton = new JButton( "7" );
304 sevenJButton.setBounds( 351, 250 , 48, 48 );
305 contentPane.add( sevenJButton );
306 keyJButtons[ KeyEvent.VK_7 ] = sevenJButton;
```

```
309
310 // set up eightJButton
311 eightJButton = new JButton( "8" );
312 eightJButton.setBounds( 399, 250 , 48, 48 );
313 contentPane.add( eightJButton );
314 keyJButtons[ KeyEvent.VK_8 ] = eightJButton;
315
316 // set up nineJButton
317 nineJButton = new JButton( "9" );
318 nineJButton.setBounds( 447 , 250 , 48, 48 );
319 contentPane.add( nineJButton );
320 keyJButtons[ KeyEvent.VK_9 ] = nineJButton;
321
322 // set up zeroJButton
323 zeroJButton = new JButton( "0" );
324 zeroJButton.setBounds( 495, 250 , 48, 48 );
325 contentPane.add( zeroJButton );
326 keyJButtons[ KeyEvent.VK_0 ] = zeroJButton;
327
328 // set up hyphenJButton
329 hyphenJButton = new JButton( "-" );
330 hyphenJButton.setBounds( 543 , 250 , 48, 48 );
331 contentPane.add( hyphenJButton );
332 keyJButtons[ KeyEvent.VK_MINUS ] = hyphenJButton;
333
334 // set up plusJButton
335 plusJButton = new JButton( "+" );
336 plusJButton.setBounds( 591, 250 , 48, 48 );
337 contentPane.add( plusJButton );
338 keyJButtons[ KeyEvent.VK_EQUALS ] = plusJButton;
339
340 // set up backspaceJButton
341 backspaceJButton = new JButton( "Backspace" );
342 backspaceJButton.setBounds( 639 , 250 , 100, 48 );
343 contentPane.add( backspaceJButton );
344 keyJButtons[ KeyEvent.VK_BACK_SPACE ] = backspaceJButton;
345
346 // set up tabJButton
347 tabJButton = new JButton( "Tab" );
348 tabJButton.setBounds( 15, 298, 75 , 48 );
349 contentPane.add( tabJButton );
350 keyJButtons[ KeyEvent.VK_TAB ] = tabJButton;
351
352 // set up qJButton
353 qJButton = new JButton( "Q" );
354 qJButton.setBounds( 90 , 298, 48, 48 );
355 contentPane.add( qJButton );
356 keyJButtons[ KeyEvent.VK_Q ] = qJButton;
357
358 // set up wJButton
359 wJButton = new JButton( "W" );
360 wJButton.setBounds( 138 , 298, 48, 48 );
361 contentPane.add( wJButton );
362 keyJButtons[ KeyEvent.VK_W ] = wJButton;
363
364 // set up eJButton
365 eJButton = new JButton( "E" );
366 eJButton.setBounds( 186 , 298, 48, 48 );
367 contentPane.add( eJButton );
368 keyJButtons[ KeyEvent.VK_E ] = eJButton;
```

```
369
370 // set up rJButton
371 rJButton = new JButton( "R" );
372 rJButton.setBounds( 234 , 298, 48, 48 );
373 contentPane.add( rJButton );
374 keyJButtons[ KeyEvent.VK_R ] = rJButton;
375
376 // set up tJButton
377 tJButton = new JButton( "T" );
378 tJButton.setBounds( 282 , 298, 48, 48 );
379 contentPane.add( tJButton );
380 keyJButtons[ KeyEvent.VK_T ] = tJButton;
381
382 // set up yJButton
383 yJButton = new JButton( "Y" );
384 yJButton.setBounds( 330, 298, 48, 48 );
385 contentPane.add( yJButton );
386 keyJButtons[ KeyEvent.VK_Y ] = yJButton;
387
388 // set up uJButton
389 uJButton = new JButton( "U" );
390 uJButton.setBounds( 378 , 298, 48, 48 );
391 contentPane.add( uJButton );
392 keyJButtons[ KeyEvent.VK_U ] = uJButton;
393
394 // set up iJButton
395 iJButton = new JButton( "I" );
396 iJButton.setBounds( 426, 298, 48, 48 );
397 contentPane.add( iJButton );
398 keyJButtons[ KeyEvent.VK_I ] = iJButton;
399
400 // set up oJButton
401 oJButton = new JButton( "O" );
402 oJButton.setBounds( 474 , 298, 48, 48 );
403 contentPane.add( oJButton );
404 keyJButtons[ KeyEvent.VK_O ] = oJButton;
405
406 // set up pJButton
407 pJButton = new JButton( "P" );
408 pJButton.setBounds( 522, 298, 48, 48 );
409 contentPane.add( pJButton );
410 keyJButtons[ KeyEvent.VK_P ] = pJButton;
411
412 // set up leftBraceJButton
413 leftBraceJButton = new JButton( "[" );
414 leftBraceJButton.setBounds( 570, 298, 48, 48 );
415 contentPane.add( leftBraceJButton );
416 keyJButtons[ KeyEvent.VK_OPEN_BRACKET ] = leftBraceJButton;
417
418 // set up rightBraceJButton
419 rightBraceJButton = new JButton( "]" );
420 rightBraceJButton.setBounds( 618, 298, 48, 48 );
421 contentPane.add( rightBraceJButton );
422 keyJButtons[ KeyEvent.VK_CLOSE_BRACKET ] = rightBraceJButton;
423
424 // set up slashJButton
425 slashJButton = new JButton( "\\" );
426 slashJButton.setBounds( 666, 298, 48, v );
427 contentPane.add( slashJButton );
428 keyJButtons[ KeyEvent.VK_BACK_SLASH ] = slashJButton;
429
```

```
430 // set up capsJButton
431 capsJButton = new JButton( "Caps" );
432 capsJButton.setBounds( 15, 346 , 75 , 48 );
433 contentPane.add( capsJButton );
434 keyJButtons[ KeyEvent.VK_CAPS_LOCK ] = capsJButton;
435
436 // set up aJButton
437 aJButton = new JButton( "A" );
438 aJButton.setBounds( 90 , 346 , 48, 48 );
439 contentPane.add( aJButton );
440 keyJButtons[ KeyEvent.VK_A ] = aJButton;
441
442 // set up sJButton
443 sJButton = new JButton( "S" );
444 sJButton.setBounds( 138 , 346 , 48, 48 );
445 contentPane.add( sJButton );
446 keyJButtons[ KeyEvent.VK_S ] = sJButton;
447
448 // set up dJButton
449 dJButton = new JButton( "D" );
450 dJButton.setBounds( 186 , 346 , 48, 48 );
451 contentPane.add( dJButton );
452 keyJButtons[ KeyEvent.VK_D ] = dJButton;
453
454 // set up fJButton
455 fJButton = new JButton( "F" );
456 fJButton.setBounds( 234 , 346 , 48, 48 );
457 contentPane.add( fJButton );
458 keyJButtons[ KeyEvent.VK_F ] = fJButton;
459
460 // set up gJButton
461 gJButton = new JButton( "G" );
462 gJButton.setBounds( 282 , 346 , 48, 48 );
463 contentPane.add( gJButton );
464 keyJButtons[ KeyEvent.VK_G ] = gJButton;
465
466 // set up hJButton
467 hJButton = new JButton( "H" );
468 hJButton.setBounds( 330, 346 , 48, 48 );
469 contentPane.add( hJButton );
470 keyJButtons[ KeyEvent.VK_H ] = hJButton;
471
472 // set up jJButton
473 jJButton = new JButton( "J" );
474 jJButton.setBounds( 378 , 346 , 48, 48 );
475 contentPane.add( jJButton );
476 keyJButtons[ KeyEvent.VK_J ] = jJButton;
477
478 // set up kJButton
479 kJButton = new JButton( "K" );
480 kJButton.setBounds( 426, 346 , 48, 48 );
481 contentPane.add( kJButton );
482 keyJButtons[ KeyEvent.VK_K ] = kJButton;
483
484 // set up lJButton
485 lJButton = new JButton( "L" );
486 lJButton.setBounds( 474 , 346 , 48, 48 );
487 contentPane.add( lJButton );
488 keyJButtons[ KeyEvent.VK_L ] = lJButton;
489
490 // set up colonJButton
```

```
491 colonJButton = new JButton( ":" );
492 colonJButton.setBounds( 522, 346 , 48, 48 );
493 contentPane.add( colonJButton );
494 keyJButtons[ KeyEvent.VK_SEMICOLON ] = colonJButton;
495
496 // set up quoteJButton
497 quoteJButton = new JButton( "\"" );
498 quoteJButton.setBounds( 570, 346 , 48, 48 );
499 contentPane.add( quoteJButton );
500 keyJButtons[ KeyEvent.VK_QUOTE ] = quoteJButton;
501
502 // set up enterJButton
503 enterJButton = new JButton( "Enter" );
504 enterJButton.setBounds( 618, 346 , 96 , 48 );
505 contentPane.add( enterJButton );
506 keyJButtons[ KeyEvent.VK_ENTER ] = enterJButton;
507
508 // set up shiftLeftJButton
509 shiftLeftJButton = new JButton( "Shift" );
510 shiftLeftJButton.setBounds( 15, 394, 100, 48 );
511 contentPane.add( shiftLeftJButton );
512 keyJButtons[ KeyEvent.VK_SHIFT ] = shiftLeftJButton;
513
514 // set up zJButton
515 zJButton = new JButton( "Z" );
516 zJButton.setBounds( 115, 394, 48, 48 );
517 contentPane.add( zJButton );
518 keyJButtons[ KeyEvent.VK_Z ] = zJButton;
519
520 // set up xJButton
521 xJButton = new JButton( "X" );
522 xJButton.setBounds( 163, 394, 48, 48 );
523 contentPane.add( xJButton );
524 keyJButtons[ KeyEvent.VK_X ] = xJButton;
525
526 // set up cJButton
527 cJButton = new JButton( "C" );
528 cJButton.setBounds( 211, 394, 48, 48 );
529 contentPane.add( cJButton );
530 keyJButtons[ KeyEvent.VK_C ] = cJButton;
531
532 // set up vJButton
533 vJButton = new JButton( "V" );
534 vJButton.setBounds( 259, 394, 48, 48 );
535 contentPane.add( vJButton );
536 keyJButtons[ KeyEvent.VK_V ] = vJButton;
537
538 // set up bJButton
539 bJButton = new JButton( "B" );
540 bJButton.setBounds( 307, 394, 48, 48 );
541 contentPane.add( bJButton );
542 keyJButtons[ KeyEvent.VK_B ] = bJButton;
543
544 // set up nJButton
545 nJButton = new JButton( "N" );
546 nJButton.setBounds( 355, 394, 48, 48 );
547 contentPane.add( nJButton );
548 keyJButtons[ KeyEvent.VK_N ] = nJButton;
549
550 // set up mJButton
551 mJButton = new JButton( "M" );
```



```
552     mJButton.setBounds( 403, 394, 48, 48 );
553     contentPane.add( mJButton );
554     keyJButtons[ KeyEvent.VK_M ] = mJButton;
555
556     // set up commaJButton
557     commaJButton = new JButton( "," );
558     commaJButton.setBounds( 451, 394, 48, 48 );
559     contentPane.add( commaJButton );
560     keyJButtons[ KeyEvent.VK_COMMA ] = commaJButton;
561
562     // set up periodJButton
563     periodJButton = new JButton( "." );
564     periodJButton.setBounds( 499, 394, 48, 48 );
565     contentPane.add( periodJButton );
566     keyJButtons[ KeyEvent.VK_PERIOD ] = periodJButton;
567
568     // set up questionJButton
569     questionJButton = new JButton( "?" );
570     questionJButton.setBounds( 547, 394, 48, 48 );
571     contentPane.add( questionJButton );
572     keyJButtons[ KeyEvent.VK_SLASH ] = questionJButton;
573
574     // set up upJButton
575     upJButton = new JButton( "^" );
576     upJButton.setBounds( 618, 394, 48, 48 );
577     contentPane.add( upJButton );
578     keyJButtons[ KeyEvent.VK_UP ] = upJButton;
579
580     // set up spaceJButton
581     spaceJButton = new JButton( " " );
582     spaceJButton.setBounds( 208, 442, 300, 48 );
583     contentPane.add( spaceJButton );
584     keyJButtons[ KeyEvent.VK_SPACE ] = spaceJButton;
585
586     // set up leftJButton
587     leftJButton = new JButton( "<" );
588     leftJButton.setBounds( 570, 442, 48, 48 );
589     contentPane.add( leftJButton );
590     keyJButtons[ KeyEvent.VK_LEFT ] = leftJButton;
591
592     // set up downJButton
593     downJButton = new JButton( "v" );
594     downJButton.setBounds( 618, 442, 48, 48 );
595     contentPane.add( downJButton );
596     keyJButtons[ KeyEvent.VK_DOWN ] = downJButton;
597
598     // set up rightJButton
599     rightJButton = new JButton( ">" );
600     rightJButton.setBounds( 666, 442, 48, 48 );
601     contentPane.add( rightJButton );
602     keyJButtons[ KeyEvent.VK_RIGHT ] = rightJButton;
603
604     // set properties of application's window
605     setTitle( "Typing Skills Developer" ); // set title bar string
606     setSize( 760, 550 ); // set window size
607     setVisible( true ); // display window
608
609 } // end method createUserInterface
610
611 // reset the color of the lastJButton
612 private void outputJTextAreaFocusLost( FocusEvent event )
```

```

613 {
614     resetColor();
615
616 } // end method outputJTextAreaFocusLost
617
618 // clear text
619 private void clearJMenuItemActionPerformed( ActionEvent event )
620 {
621     outputJTextArea.setText( "" );
622
623 } // end method clearJMenuItemActionPerformed
624
625 // highlight JButton passed as argument
626 private void changeColor( JButton changeJButton )
627 {
628     if ( changeJButton != null )
629     {
630         resetColor();
631         changeJButton.setBackground( Color.YELLOW );
632         lastJButton = changeJButton;
633     }
634
635 } // end method changeColor
636
637 // changes lastJButton's color back to default
638 private void resetColor()
639 {
640     if ( lastJButton != null )
641     {
642         lastJButton.setBackground( this .getBackground() );
643     }
644
645 } // end method resetColor
646
647 // change text color when user selects Colors... JMenuItem
648 private void colorJMenuItemActionPerformed( ActionEvent event )
649 {
650     Color foregroundColor = JColorChooser.showDialog(      显示JColorChooser
651         this , "Choose a color" , Color.BLACK );
652
653     // if the user selected a color
654     if ( foregroundColor != null )
655     {
656         // set foreground color of outputJTextArea
657         outputJTextArea.setForeground( foregroundColor );  将文本改为用户所选颜色
658     }
659
660 } // end method colorMenuItemActionPerformed
661
662 // change font when user selects an item from Style submenu
663 private void styleMenuItemsStateChanged( ItemEvent event )
664 {
665     int style = Font.PLAIN;      初始化一个用来保存字形的变量
666
667     // check for bold selection
668     if ( styleMenuItems[ 0 ].isSelected() )      测试是否选取Bold菜单选项
669     {
670         style += Font.BOLD;
671     }
672

```

```

673 // check for italic selection
674 if ( styleMenuItems[ 1 ].isSelected() )           测试是否选取 Italic 菜单选项
675 {
676     style += Font.ITALIC;
677 }
678
679 // create a new Font with the specified style
680 outputFont = new Font( outputFont.getName(), style,      新建一个具有
681     outputFont.getSize() );                             指定字形的 Font
682
683 outputJTextArea.setFont( outputFont ); // set the font      设置 outputJTextArea 中文本的字体
684
685 } // end method styleMenuItemsStateChanged
686
687 // change font size when user selects an item from size submenu
688 private void sizeMenuItemsActionPerformed((ActionEvent event) )
689 {
690     JRadioButtonMenuItem sizeMenuItem =                存储 JRadioButton-
691         ( JRadioButtonMenuItem ) event.getSource();    MenuItem 的引用
692
693     // create a new Font with the specified size
694     outputFont = new Font( outputFont.getName(),
695         outputFont.getStyle(),                          新建一个具有指定字号的 Font
696         Integer.parseInt( sizeMenuItem.getText() ) );
697
698     outputJTextArea.setFont( outputFont ); // set the font
699
700 } // end method sizeMenuItemsActionPerformed
701
702 // highlight corresponding JButton when a key is pressed
703 private void outputJTextAreaKeyPressed( KeyEvent event )
704 {
705     // get the key code for this event
706     int buttonIndex = event.getKeyCode(); 利用getKeyCode方法确定被按下的是哪一个按键
707
708     // change the color of the associated JButton
709     changeColor( keyJButtons[ buttonIndex ] );
710
711 } // end method outputJTextAreaKeyPressed
712
713 // reset the color of the pressed key's JButton
714 private void outputJTextAreaKeyReleased( KeyEvent event )
715 {
716     resetColor();                                       将lastJButton的背景色改为默认颜色
717
718 } // end method outputJTextAreaKeyReleased
719
720 // main method
721 public static void main( String[] args )
722 {
723     TypingApplication application = new TypingApplication();
724     application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
725
726 } // end method main
727
728 } // end class TypingApplication

```

图 22.31 打字训练器应用程序源代码

自测题

1. _____ 对话框允许用户选择颜色。
a) ColorChooser b) JColorChooser c) ChooseColor d) JChooseColor
2. 使用 _____ 方法可返回所对应的事件源。
a) getSource b) getEventSource c) eventSource d) source

答案: 1) b 2) a

22.5 小结

在本教程中,学习了与键盘事件有关的内容,懂得如何处理键盘按键按下时所产生的事件(利用 keyPressed 事件处理程序)。之后,还学习了如何使用 keyReleased 事件处理程序,完成因释放某个按键时所产生的事件。

通过在打字训练器应用程序中添加 JMenu,在不影响 GUI 中已添加组件的前提下,对应用程序的功能进行了扩充。这些 JMenu 应放置在一个 JMenuBar 中。随后,我们通过向 JMenu 中添加分隔条,完成对菜单的组织管理。同时,还学习了如何显示 JColorChooser 对话框,使用户对 JTextArea 中的文本指明所需要的颜色。

在下一个教程中,将学习如何管理字符串,并通过 String 类的方法创建一个屏幕抓取应用程序,利用该应用程序实现对 Web 页面中特定信息的查找。

技术小结

执行键盘中某按键被按下时的代码

- 使用 keyPressed 事件处理程序。
- 使用 KeyEvent 的 getKeyCode 方法确定哪一个按键被按下。
- 在每一个 case 中比较所按按键与虚键码的值。

执行某按键被释放时的代码

- 使用 keyReleased 事件处理程序。

将 JMenu 添加到应用程序中

- 创建一个能容纳所有菜单的 JMenuBar。
- 创建 JMenu 并利用 JMenuBar 类的 add 方法将其添加到 JMenuBar 中。
- 创建 JMenuItem 并利用 JMenu 类的 add 方法将其添加到 JMenu 中。
- 为便于访问,使用 setMnemonic 方法为 JMenu 和 JMenuItem 添加助记符。

为应用程序添加颜色对话框

- 使用 JColorChooser 类的 showDialog 方法为用户提供一个允许选择颜色的对话框。

将 JCheckBoxMenuItem 添加到 JMenu 中

- 创建一个 JCheckBoxMenuItem。
- 使用 JMenu 类的 add 方法将 JCheckBoxMenuItem 添加到 JMenu 中。

将 JRadioButtonMenuItem 添加到 JMenu 中

- 创建 JRadioButtonMenuItem。
- 使用 JMenu 类的 add 方法将 JRadioButtonMenuItem 添加到 JMenu 中。

将 JMenuItem 添加到 JMenu 中

- 创建一个 JMenuItem。
- 使用 JMenu 类的 add 方法将 JMenuItem 添加到 JMenu 中。

关键术语

addKeyListener 方法 为某个事件源（如一个 JTextArea）注册事件监听器。

JMenu 类的 add 方法 用于将 JMenu, JMenuItem, JCheckBoxMenuItem 或者 JRadioButtonMenuItem 添加到 JMenu 中。

JMenuBar 类的 add 方法 能够将 JMenu 添加到 JMenuBar 中。

JMenu 类的 addSeparator 方法 为 JMenu 添加分隔条。

Font 类 创建并管理字体的一个类。

常量 Font.BOLD 表示粗体字。

常量 Font.ITALIC 表示斜体字。

常量 Font.PLAIN 表示常规字。

KeyEvent 类的 getKeyCode 方法 返回用户交互时所按按键的虚键码。

Font 类的 getName 方法 返回字体名称。

Font 类的 getSize 方法 返回字号大小。

ActionEvent 类的 getSource 方法 返回事件产生的 GUI 组件（也称事件源）。

Font 类的 getStyle 方法 返回字形。

包 java.awt.event 提供了有关事件处理（如键盘事件和鼠标事件等）所需的接口和类。

JCheckBoxMenuItem 组件 一种特定类型的菜单选项，其左侧拥有一个复选框，允许用户查看当前选择的选项。

JColorChooser 组件 为用户提供选择颜色选项的一个对话框。

JMenu 组件 能够添加到 JMenuBar 中并包含其他 JMenu 和 JMenuItem 的菜单。

JMenuBar 组件 用来容纳 JMenu 并位于应用程序顶端的一个条状组件。

JMenuItem 组件 菜单中的选项。

JRadioButtonMenuItem 组件 一种特定类型的菜单选项，其左侧拥有一个单选框，允许用户查看当前选项的选择情况。

键盘事件 当按下、释放，或者按下然后释放键盘上的某个按键时所产生的的一种事件。

KeyEvent 当用户按下、释放某个按键时生成的一种事件。

KeyEvent.KEY_LAST 虚键码常量中的最大值。

KeyListener 接口 针对键盘事件而声明了一些相关事件处理程序首部的接口。

keyPressed 事件处理程序 按下某个按键时被调用的方法。

keyReleased 事件处理程序 释放某个按键时被调用的方法。

keyTyped 事件处理程序 定义在 KeyListener 中的方法接口。此方法在按下然后释放某个按键时被调用。

菜单 将 GUI 应用程序中相关命令组织在一起的一个组件。菜单在 GUI 中属于一个独立部分，它们在不干扰其他 GUI 的前提下可将多个命令组织在一起。在 Java 中，菜单是由 JMenu 组件来创建的。

菜单条 位于应用程序顶部，内含菜单的一个条状组件。在 Java 中，菜单条是由 JMenuBar 组件来创建的。

菜单选项 位于菜单内的选项，一旦选取，会导致应用程序执行某个特定操作。在 Java 中，菜单选项是由 JMenuItem 组件来创建的。

分隔条 位于菜单内以分隔不同的菜单选项。

JTextArea 类的 setFont 方法 设置 JTextArea 中文本的字体。

JTextArea 类的 setForeground 方法 设置 JTextArea 的前景色。

JFrame 类的 setJMenuBar 方法 设置 JFrame 中的 JMenuBar。

- JMenu 类的 setMnemonic 方法 为 JMenu 设置键盘快捷键。
- JMenuItem 类的 setMnemonic 为 JMenuItem 设置键盘快捷键。
- JColorChooser 类的 showDialog 方法 显示一个颜色对话框。
- 子菜单 位于另一个菜单内的菜单。
- 顶级菜单 直接添加在菜单条中的菜单。

GUI 设计导航

菜单

- 菜单选项中的文本应使用书名大写形式。
- 如果点击某个菜单选项时会打开一个对话框，则在该菜单选项文本的后面添加一个省略号。
- 利用分隔条分组管理 JMenu 中的多个 JMenuItem。

JCheckBoxMenuItem

- JCheckBoxMenuItem 中的文本应具有描述性质且尽可能地简短，同时还应使用书名大写形式。
- 当用户需要从菜单中选择多个选项时，可以考虑使用 JCheckBoxMenuItem。

JRadioButtonMenuItem

- 当用户需要从菜单中选择惟一的一个选项时，可以考虑使用 JRadioButtonMenuItem。

Java 类库索引

ActionEvent 该类表示传递给 actionPerformed 事件处理程序的参数。

- 方法
 - getSource 返回事件源。

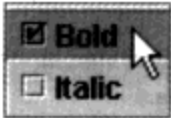
Font 此类表示一个由名称、字形和字号所确定的惟一字体。

- 构造方法
 - Font 接收 3 个参数：字体名称、字形和字号。

```
outputFont = new Font( outputFont.getName(), Font.BOLD, 16 );
```
- 常量
 - Font.BOLD 表示粗体字。
 - Font.ITALIC 表示斜体字。
 - Font.PLAIN 表示常规字。
- 方法
 - getName 将字体名称以字符串形式返回。
 - getSize 将字号大小以 int 值形式返回。
 - getStyle 将字形以 int 值形式返回。

JCheckBoxMenuItem 该组件表示一个菜单对象，其左侧拥有一个可告知用户是否已选取该对象的 JCheckBox 组件。

- 运行



- 方法
 - getText 返回 JCheckBoxMenuItem 上的文本。
 - isSelected 返回组件是否被选取。

JColorChooser 该组件允许用户选择某种颜色。

● 方法

showDialog 显示一个 JColorChooser 对话框。

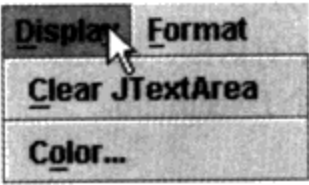
JMenu 该组件将应用程序中的相关菜单选项集合在一起。

● 构造方法

JMenu 接收一个指明 JMenu 名称的参数。

```
displayJMenu = new JMenu( "Display" );
```

● 运行

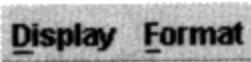


● 方法

- add 将菜单或菜单选项添加到一个菜单中。
- addSeparator 在菜单中加入一个分隔条。
- setMnemonic 为菜单设置键盘快捷键。

JMenuBar 该组件允许将菜单添加到应用程序中。

● 运行

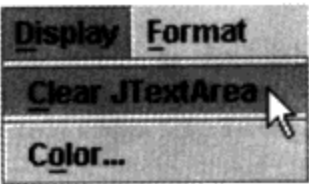


● 方法

- add 将菜单添加到菜单条中。

JMenuItem 该组件代表一个添加到 JMenu 中的菜单选项。

● 运行

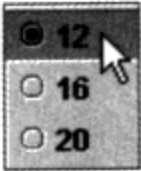


● 方法

- setMnemonic 设置 JMenuItem 的键盘快捷键。

JRadioButtonMenuItem 该组件表示一个菜单对象,其左侧拥有一个告知用户是否已选择该对象的单选框。

● 运行

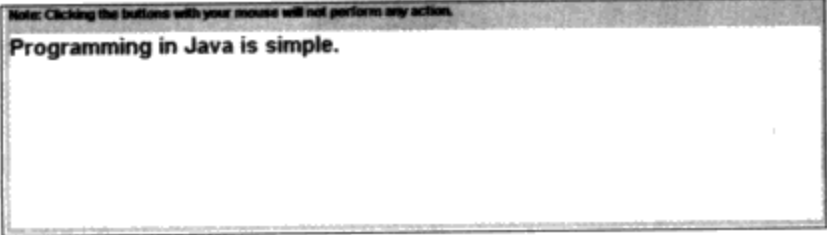


● 方法

- getText 返回 JRadioButtonMenuItem 上的文本。

JTextArea 该组件用于显示从键盘中输入的数据。

● 运行



● 方法

addKeyListener 将一个 **KeyListener** 对象添加到 **JTextArea** 中。

append 将文本添加到 **JTextArea** 中。

grabFocus 将应用程序的焦点转移到 **JTextArea** 上。

setBounds 设置 **JTextArea** 的位置和大小。

setEditable 设置 **JTextArea** 中文本的可编辑性 (**true/false**)。

setEnabled 设置 **enabled** 属性, 确定 **JTextArea** 是否能响应用户的操作 (**true/false**)。

setFont 设置 **JTextArea** 中文本的字体。

setForeground 设置 **JTextArea** 的前景色。

setHorizontalAlignment 指定 **JTextArea** 中文本的对齐方式 (**JTextArea.LEFT**, **JTextArea.CENTER**, **JTextArea.RIGHT**)。

setText 设置 **JTextArea** 中要显示的文本。

KeyEvent 该类表示当某个按键被按下、释放, 或者按下然后释放时所生成的事件对象。

● 常量

KeyEvent.KEY_LAST **KeyEvent** 类中一个最大虚键码。

KeyEvent.VK_A 代表键盘上的 A 键。

KeyEvent.VK_B 代表键盘上的 B 键。

● 方法

getKeyCode 返回用户交互时所按按键的虚键码。

KeyListener 此接口针对键盘事件定义了三个事件处理程序。

● 事件处理程序

keyPressed 当按下某个按键时调用。

keyReleased 当释放某个按键时调用。

keyTyped 当按下然后释放某个字符按键时调用。

习题

选择题

- 22.1 **KeyEvent** 中的常量 _____ 代表最大的一个虚键码。
a) **MAXIMUM** b) **KEY_FINAL** c) **VK_MAX** d) **KEY_LAST**
- 22.2 _____ 是一个菜单容器。
a) 键盘事件 b) 菜单条 c) 分隔条 d) 菜单选项
- 22.3 添加到 **JMenu** 中以创建子菜单的 GUI 组件为 _____。
a) **JSubmenu** b) **JMenuItem** c) **JMenu** d) **JMenuBar**
- 22.4 **addSeparator** 方法能够为菜单添加一个分隔 _____。
a) 条 b) 助记符 c) 子菜单 d) 菜单选项
- 22.5 _____ 能够分组管理应用程序中的相关命令。
a) 分隔条 b) 热键 c) 菜单 d) 边界指示条
- 22.6 **KeyEvent** 的 _____ 方法可返回用户所按按键。
a) **keys** b) **KeyCode** c) **add** d) **getKeyCode**
- 22.7 当用户按下某个按键时, 会调用 _____ 事件处理程序。
a) **keyDown** b) **keyReleased** c) **keyPressed** d) 以上答案都不对
- 22.8 使用 **JTextArea** 的 _____ 方法来设置 **JTextArea** 中文本的颜色。
a) **setForeground** b) **setTextColor** c) **setBackground** d) **setColor**
- 22.9 以下哪一个方法用来设置菜单选项的助记符?
a) **setFont** b) **setMnemonic** c) **setMenuMnemonic** d) **setText**

22.10 当用户一次只能选择惟一的一个菜单选项时, 应考虑使用_____。

- a) 子菜单 b) JRadioButtonMenuItem c) JColorChooser d) JCheckBoxMenuItem

练习题

22.11 (改进的库存清单应用程序)改进教程4中曾经开发的库存清单应用程序, 经过改进的应用程序可防止用户键入非数字数据。通过使用键盘事件, 只允许用户按下数字按键(0~9)、左右方向键、退格键以及回车键。如果按下其他按键, 显示一个 JOptionPane 并告知用户需输入数字(参见图 22.32)。

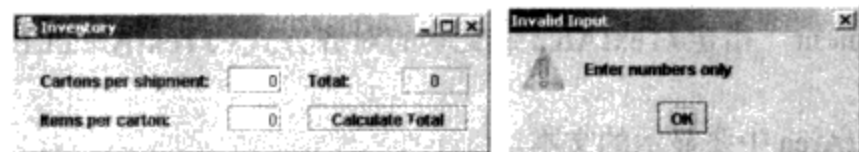


图 22.32 改进的库存清单应用程序的 GUI

- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial22\Exercises\InventoryEnhanced 目录复制到 C:\SimplyJava 目录中。
 - b) 打开模板文件 在自己的文本编辑器中打开 Inventory.java 文件。
 - c) 添加一条 switch 语句 在第 132 行(位于 cartonsJTextFieldKeyPressed 方法内), 编写一条 switch 语句, 判定所按按键是否为数字键、左右方向键、退格键或者回车键(利用 KeyEvent 常量 VK_0, VK_1, VK_2, VK_3, VK_4, VK_5, VK_6, VK_7, VK_8, VK_9, VK_LEFT, VK_RIGHT, VK_BACK_SPACE, VK_ENTER)。如果这些按键被按下, 利用 break 语句继续应用程序的执行。
 - d) 添加 default 语句 在该 switch 语句的结尾处, 通过添加 default 语句判定是否存在其他非有效按键。如果存在非有效按键, 则显示一个 JOptionPane 并告知用户需输入一个数字。同时, 将 cartonsJTextField 中的文本恢复为 "0"。
 - e) 再添加一条 switch 语句 在 itemsJTextFieldKeyPressed 方法内, 编写一条 switch 语句, 判定所按按键是否为数字键、左右方向键、退格键或者是回车键[使用与步骤(c)中一样 KeyEvent 常量]。如果这些按键被按下, 利用 break 语句继续应用程序的执行。
 - f) 添加 default 语句 在该 switch 语句的结尾处, 通过添加 default 语句判定是否存在其他非有效按键。如果存在非有效按键, 则显示一个 JOptionPane 并告知用户需输入一个数字。同时, 将 itemsJTextField 中的文本恢复为 "0"。
 - g) 保存应用程序 保存修改后的源代码文件。
 - h) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\InventoryEnhanced 进入到当前工作目录中。
 - i) 编译模板应用程序 键入 javac Inventory.java 对该应用程序进行编译。
 - j) 运行完成后的应用程序 若能正确编译应用程序, 输入 java Inventory 来运行它。可以在每一个 JTextArea 中按下一些有效按键和无效按键, 实现对应用程序的测试。当按下无效按键时, 会出现一个 JOptionPane。
 - k) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
 - l) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。
- 22.12 (弹力球应用程序)编写一个游戏应用程序, 此游戏目标是阻止一个弹力球下落到应用程序的底边界上。用户按下 S 按键时, 游戏开始, 此时, 一个蓝色小球将在应用程序的上边界、左边界以及右边界(“墙”)来回弹动。应用程序的下边界处还应有一个水平条, 作为球拍来使用, 阻止小球撞击到下边界上(小球可在球拍上重新弹起, 但不能落到下边界上)。用户可以利用左右方向键移动球拍。如果小球撞到球拍, 小球则被弹起, 游戏继续进行。如果小球落到下边界上, 则游戏结束。凡与几何处理相关的计算以及所需的 GUI(如图 22.33 所示), 已在模板中提供给了读者。

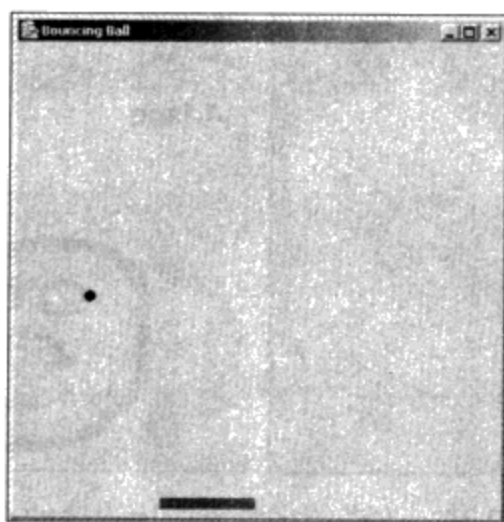


图 22.33 弹力球应用程序

- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial22\Exercises\BouncingBall 目录复制到 C:\SimplyJava 目录中。
 - b) 打开模板文件 在自己的文本编辑器中打开 BouncingBall.java 文件。
 - c) 编写用于启动游戏的代码 在第 131 行 (位于 bouncingBallKeyPressed 方法的内部), 编写一条 if 语句, 判断 S 键是否被按下 (需使用 KeyEvent 的常量 VK_S)。在 if 语句的内部, 调用 ballTimer 的 start 方法启动 ballTimer, 此变量在模板中已作为实例变量声明为一个 Timer。
 - d) 插入用于将球拍向左移动的代码 在步骤(c)所添加的 if 语句后面, 加入一条 else if 语句, 判断用户是否按下左方向键, 以及球拍的水平位置 (存储在 rectX 中) 是否大于 10。读者需要用到 KeyEvent 的常量 VK_LEFT。如果球拍的水平位置等于 10, 表明其左边缘刚好接触到应用程序的左边界, 因此球拍将不允许再向左移动。在 else if 语句的内部, 将球拍的水平位置减去 10 个像素。
 - e) 插入可将球拍向右移动的代码 在步骤(d)所添加的 else if 语句的后面, 再加入一条 else if 语句, 判断用户是否按下右方向键, 以及球拍的水平位置 (存储在 rectX 中) 是否小于 400 减球拍宽度 (存储在 rectWidth 中)。在 else if 语句的内部, 将球拍的水平位置移动 10 个像素。
 - f) 保存应用程序 保存修改后的源代码文件。
 - g) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\BouncingBall 进入到当前工作目录中。
 - h) 编译模板应用程序 键入 javac BouncingBall.java 对应用程序进行编译。
 - i) 运行完成后的应用程序 若能正确编译应用程序, 键入 java BouncingBall 来运行它。为测试应用程序, 当按下 S 键时应确保出现一个不断移动的小球并且当小球撞击到球拍上时能够继续移动, 而落到球拍以下时小球将停止移动。
 - j) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
 - k) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。
- 22.13 (改进的小画家应用程序) 改进教程 21 中曾开发的小画家应用程序, 为其添加一些菜单, 使用户通过操作菜单为已绘制的椭圆选择大小、颜色并实现 JFrame 颜色的选择 (如图 22.34) (利用菜单替换以前的 JRadioButton)。同时, 再添加一个 JTextArea, 允许用户输入与所绘图形相关的描述。用户可利用菜单对 JTextArea 中文本的字形、颜色以及 JTextArea 的背景色进行设置。
- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial22\Exercises\PainterEnhanced 目录复制到 C:\SimplyJava 目录中。
 - b) 打开模板文件 在自己的文本编辑器中打开 Painter.java 文件。
 - c) 创建 paintJMenu 在第 75 行, 添加创建 paintJMenu 的代码。完成此项工作应创建一个新的 JMenu 并为其赋予 paintJMenu。接着, 将 paintJMenu 的文本设置为 "Paint", 再将 paintJMenu 的助记符设置为 P 键 (需使用 KeyEvent 的常量 VK_P)。之后, 将 paintJMenu 添加到 painterJMenuBar 中。

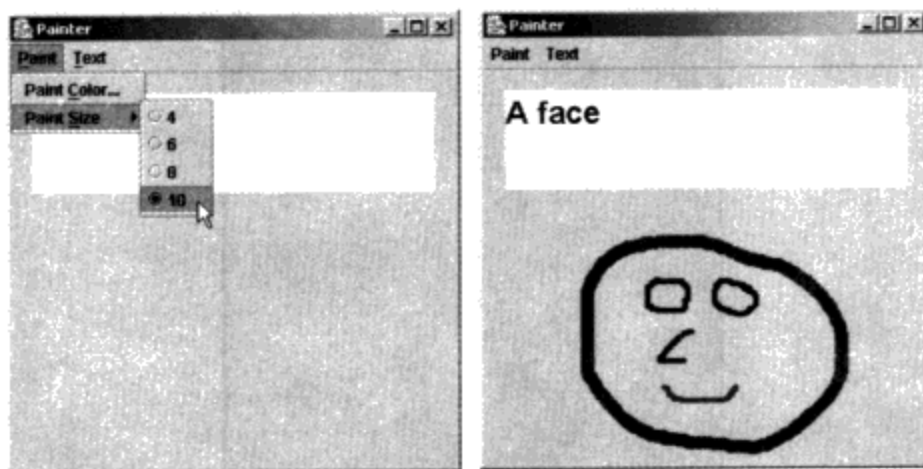


图 22.34 改进的小画家应用程序的 GUI

- d) **创建 textJMenu** 在创建 textColorJMenuItem 代码的前面，添加用于创建 textJMenu 的代码。完成此项工作应创建一个新的 JMenu 并为其赋予 textJMenu。接着，将 textJMenu 的文本设置为 "Text"，再将 textJMenu 的助记符设置为 T 键（需使用 KeyEvent 的常量 VK_T）。之后，将 textJMenu 添加到 painterJMenuBar 中。
- e) **允许用户选择图形颜色** 在 paintColorJMenuItemActionPerformed 方法内，声明一个 Color 类型的局部变量 paintColor 并赋予调用 JColorChooser 类的 showDialog 方法所返回的值。之后，添加一条 if 语句判断 paintColor 是否不等于 null。在 if 语句体内，利用 setColor 方法设置 myPainterJPanel 的颜色并为该方法传递参数 paintColor。
- f) **允许用户选择字号** 在 fontSizeItemsActionPerformed 方法内，调用(ActionEvent 事件对象的 getSource 方法。将返回值造型为 JRadioButtonMenuItem 类型并赋值给一个名为 sizeMenuItem 的局部变量。注意，变量 sizeMenuItem 必须为 JRadioButtonMenuItem 类型。然后，创建一种新的字体并把它赋值给实例变量 displayFont。传递至 Font 类构造方法中的前两个参数应为 displayFont 字体的名称和字形。最后一个参数为等价于 sizeMenuItem 上文本值大小的 int 型整数。读者需调用 sizeMenuItem 的 getText 方法并将返回值赋值给 Integer.parseInt 方法。最后，将 displayJTextArea 的字体设置为 displayFont。
- g) **保存应用程序** 保存修改后的源代码文件。
- h) **打开命令提示符窗口改变目录** 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\PainterEnhanced 进入到当前工作目录中。
- i) **编译模板应用程序** 键入 javac Painter.java 对该应用程序进行编译。
- j) **运行完成后的应用程序** 若能正确编译应用程序，通过输入 java Painter 来运行它。通过创建具有不同颜色的图形对该应用程序进行测试，并在 JTextArea 中添加一条相应的描述性文本。确保能够使用添加到应用程序中的菜单对已输入文本的字体颜色和字形进行更改。
- k) **关闭应用程序** 点击关闭按钮关闭正在运行的应用程序。
- l) **关闭命令提示符窗口** 点击命令提示符窗口的关闭按钮将其关闭。

22.14 (说出这段代码的作用) 以下代码的执行结果是什么？假设 outputJTextArea 属于 JTextArea 类型。

```

1 private void colorJMenuItemActionPerformed( ActionPerformed event )
2 {
3     Color backgroundColor = JColorChooser.showDialog(
4         null, "Choose a color" , Color.BLACK );
5
6     if ( backgroundColor != null )
7     {
8         outputJTextArea.setBackground( backgroundColor );
9     }
10
11 } // end method colorJMenuItemActionPerformed

```

22.15 (找出代码中的错误) 下面这段代码将在按下 B 键时, 自增变量 bCount。请找出代码中的错误, 假设 bCount 和 outputJTextArea 分别为 int 类型和 JTextArea 类型。

```

1 private void outputJTextAreaKeyPressed( KeyEvent event)
2 {
3     // receive code for key pressed by user
4     switch ( event.KeyCode() )
5     {
6         case VK_B : // B key
7             bCount++;
8             break ;
9     }
10 } // end event handler outputJTextAreaKeyPressed

```

挑战题

22.16 (Dvorak 键盘应用程序) 创建一个用来模拟 Dvorak 键盘字母的应用程序 (相关知识请参考网站 <http://www.mwbrooks.com/dvorak/>, <http://www.mit.edu/people/jcb/Dvorak/>, <http://www.cse.ogi.edu/~dylan/dvorak/dvorak.html>)。Dvorak 键盘的输入速度要比现在广泛使用的 QWERTY 键盘更快, 因为它将最常用的按键放在了最容易接触到的位置。利用键盘事件创建这样一个应用程序, 该应用程序类似打字训练器应用程序, 它使用 Dvorak 键盘替代原先的标准键盘 (参见图 22.35)。在使用过程中, 需要将正确的 Dvorak 按键高亮显示在对应的虚键盘上并将字符显示在 JTextArea 内。Dvorak 键盘上的按键与字符的映射关系如下所示:

- 在第一排, Dvorak 键盘上的 P 键对应标准键盘上的 R 键, L 键对应标准键盘上的 P 键。
- 在中间一排, Dvorak 键盘上的 A 键同标准键盘上的 A 键相同, S 键对应标准键盘上的分号 (;) 键。
- 在最下面一排, Dvorak 键盘上的 Q 键对应标准键盘上的 X 键, Z 键对应标准键盘上的问号 (?) 键。
- 其余 Dvorak 键盘上的按键可参照图 22.35。

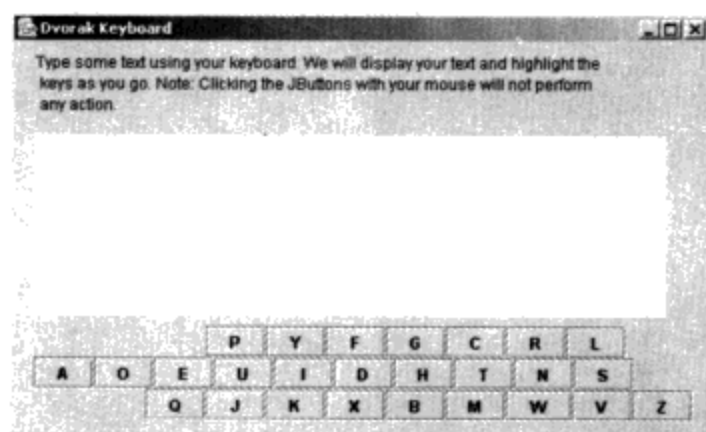


图 22.35 Dvorak 键盘应用程序的 GUI

- 将模板复制到工作目录中 将 C:\Examples\Tutorial22\Exercises\DvorakKeyboard 目录复制到 C:\SimplyJava 目录中。
- 打开模板文件 在自己的文本编辑器中打开 DvorakKeyboard.java 文件。
- 加入一个 KeyListener 从第 72 行开始, 针对 displayJTextArea 调用方法 addKeyListener, 并为该方法传递一个实现 KeyListener 接口的匿名内部类。在匿名内部类的内部, 定义三个事件处理程序—— keyPressed, keyReleased 和 keyTyped。在 keyPressed 事件处理程序中, 调用方法 displayJTextAreaKeyPressed 并为其传递一个 KeyEvent 事件对象。在 keyReleased 事件处理程序中, 调用 displayJTextAreaKeyReleased 方法, 同样也为其传递一个 KeyEvent 事件对象。keyTyped 事件处理程序则保持为空。
- 定义 displayJTextAreaKeyPressed 方法 在 createUserInterface 方法的后面, 应用程序已定义了用来处理相关 JButton 的 changeColor 方法和 resetColor 方法。在 resetColor 方法的后面, 需定

义一个 `displayJTextAreaKeyPressed` 方法。在 `displayJTextAreaKeyPressed` 方法的内部,取得与所按按键相对应的那个 `JButton` 的引用。需要使用 `KeyEvent` 对象的 `getKeyCode` 方法获得相应的虚键码,并通过它访问 `keyJButtons` 数组(已在模板中声明)。利用一条 `if` 语句判断所返回的 `JButton` 的引用是否为 `null`。如果该引用不为 `null`,则将此 `JButton` 上的文本添加到 `String` 型实例变量 `display` (已在模板中声明)中,该字符串用于存储已输入的文本。通过设置 `displayJTextArea` 的文本,显示已输入的内容。最后,调用 `changeColor` 方法并将该 `JButton` 作为参数传递给它。

- e) 定义 `displayJTextAreaKeyReleased` 方法 在 `displayJTextAreaKeyPressed` 方法的后面,定义 `displayJTextAreaKeyReleased` 方法。此方法需调用 `resetColor` 方法。
- f) 保存应用程序 保存修改后的源代码文件。
- g) 打开命令提示符窗口改变目录 选择 `Start → Programs → Accessories → Command Prompt` 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\DvorakKeyboard` 进入到当前工作目录中。
- h) 编译模板应用程序 键入 `javac DvorakKeyboard.java` 对应用程序进行编译。
- i) 运行完成后的应用程序。若能正确编译应用程序,键入 `java DvorakKeyboard` 来运行它。利用 `Dvorak` 键盘输入一些字符,从而完成对应用程序的测试。输入的时候,确保能够高亮显示正确的 `JButton`,其结果也能出现在 `JTextArea` 中。
- j) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- k) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。



教程 23 屏幕抓取应用程序

介绍字符串处理技术

教学目标

在本教程中，读者将学到以下内容：

- 创建并使用 String 对象
- 学习使用 String 类的属性及其方法
- 在字符串中查找子字符串
- 从字符串中提取子字符串
- 在字符串中替换子字符串

本教程介绍 Java 的字符串处理功能。这里所给出的一些技术，完全可以应用到与文本处理相关的应用程序当中。本书曾在前面的教程中介绍过 String 类及其一些方法的使用。在本教程中，读者将进一步学习如何查找字符串、如何从字符串中检索出子字符串，以及如何替换字符串中的子字符串等内容。读者将利用字符串的处理功能创建一个应用程序，管理含 HTML（HyperText Markup Language，超文本标记语言）字符串内容的文本。HTML 是一个用来描述 Web 内容的技术。从组成 Web 页面的 HTML 中提取出所需要信息的过程，被称为屏幕抓取，而我们所熟知的屏幕抓取器便是类似于这样的一个用于从 Web 网站上截取相应数据的工具。执行屏幕抓取的应用程序，可以从 Web 页面上获取特定的信息，如天气情况、股票价格，等等，这些信息能够很方便地应用到计算机程序中并加以利用。在本教程中，读者将创建这样一个简单的屏幕抓取应用程序。

23.1 探试屏幕抓取应用程序

此应用程序需满足下列需求：

应用程序需求分析

一家欧洲在线拍卖所，希望通过接纳来自美国的投标人扩展其商业范围，但拍卖所当前所有的 Web 网页只能显示欧元而不能显示美元。因此，拍卖公司希望为来自美国的投标人，按照美元价格的形式在 Web 页面上显示出所拍卖物品的价格。这些新创建的 Web 页面，都是通过原有的 Web 页面上使用屏幕抓取技术而生成的。现开发这样一个原型应用程序，实现屏幕抓取功能的测试。此应用程序需要实现查找一个简单的 HTML 字符串的功能，并能够提取出某个指定拍卖物品的价格信息。为完成测试，还需提供一个包含 HTML 内所有竞拍物品的 JComboBox。经选择后的物品价格会被随即转换成美元的形式，为此还应向用户提供一个用以输入当前转换汇率的 JTextField。该应用程序通过查询，取出所选物品的欧元价格，然后转换成美元价格并最终显示出来。

屏幕抓取应用程序需要查找以 HTML 字符串形式显示的某个指定拍卖物品的名称，而要查找的拍卖物品是用户通过 JComboBox 来选取的。在选取完一件拍卖物品以后，该应用程序会随即执

提取拍卖物品的价格
将该拍卖物品的价格从欧元转换成美元
在 JTextField 中显示该拍卖物品的价格

在探试完屏幕抓取应用程序并研究了它的伪代码表示后，我们使用一张 ACE 表，帮助读者最终将这个伪代码转换成 Java 实现。图 23.5 中列出了该应用程序中相应的操作、组件以及事件，帮助读者最终完成属于自己的这一应用程序。


	操作	组件	事件
	标记应用程序中的组件	itemJLabel rateJLabel priceJLabel sourceJLabel	当应用程序运行时
	在 JTextArea 中显示包含所需物品价格的 HTML	sourceJTextArea	
	从 JComboBox 中获取所选择的拍卖物品	searchJButton	当用户点击 Search JButton 时
	根据用户选择的拍卖物品查找其相应的 HTML 内容	itemJComboBox	
	提取出拍卖物品的价格	htmlText(String)	
	将该拍卖物品的价格从欧元转换成美元	htmlText(String)	
	在 JTextField 中显示该物品的价格	price(String)	
		priceJTextField	

图 23.5 屏幕抓取应用程序的 ACE 表

截至目前，我们已经分析了屏幕抓取应用程序中所使用的组件，下面将学习一些在创建该应用程序时用到的字符串处理方法。

23.4 在字符串中定位子字符串

在许多应用程序中，经常需要在一个字符串中查找某个字符或者是一组字符。比如，对于一个字处理软件来说，可能会为用户提供查找相应文档内容的功能。String 类中有一些方法，能够帮助开发人员编写查找某个字符串中的特定子字符串的功能。所谓子字符串实际是由原字符串中的一部分或者是其全部而构成的一个字符序列。例如，"lo" 是字符串 "hello" 的一个子字符串，而像 "ll", "he", "e" 甚至 "hello" 本身也都属于它的子字符串。下面，我们将利用一个 indexOf 方法查找某个字符串中的子字符串。

定位所选拍卖物品的价格

- 1. 将模板复制到工作目录中 将 C:\Examples\Tutorial23\TemplateApplication\ScreenScraping 目录复制到 C:\SimplyJava 目录中。
- 2. 打开屏幕抓取应用程序的模板文件 在自己的文本编辑器中打开模板文件 ScreenScraping.java。
- 3. 查看应用程序中的 HTML Source:JTextArea 中显示的 HTML 的内容实际为图 23.6 中第 34 行至第 42 行上的一个字符串。这里，我们并不要求读者完全理解 HTML 的含义。在本教程中，我们也只是简单地描述一下图 23.6 中的这个 HTML。读者会在教程 30 中学到更多有关 HTML 的知识，教程 30 中将在一个基于 Web 的书店应用程序中使用到 HTML。此外，与随书附带的 CD-ROM 中，还为读者额外地提供了两个章节，届时将详细介绍有关 HTML 的使用。下图中的这个 HTML，包含了三件要拍卖物品的名称和价格。第 36 行至第 37 行中所显示的这部分 HTML，为第一件拍卖物品的价格与名称。其中，第 36 行是将该拍卖物品的名称设定为 "Antique Rocking Chair"，第 37 行则是将其价格设定为 "€ 82.67"。字符串 "€" 是一个代表欧元符号 (€) 的 HTML 表示，它将出现在该应用程序中的 HTML

字符串内的每一个价格的前面。我们尚且不必担心那些位于拍卖物品名称和价格周围的文本——大多数的这些文本只是用来指明该拍卖物品的名称和价格是如何在一个Web页面中进行显示的。在本应用程序中,只需要关注每件拍卖物品的名称以及欧元价格这两项内容。第38行至第39行所显示的HTML部分代表第二件拍卖物品的名称和价格,同样,第40行至第41行所显示的HTML部分代表第三件拍卖物品的名称及其价格。



图 23.6 含拍卖物品价格的 HTML

4. 定位所选定拍卖物品的名称 将图 23.7 中第 150 行至第 153 行添加到方法 searchJButtonActionPerformed 中。第 151 行至第 152 行用于获取用户所选择的拍卖物品并将其存储在 String 型的 selectedItem 中。第 153 行通过调用 String 的 indexOf 方法,在存有 HTML 的 htmlText 中定位出所选拍卖物品名称首次出现时的位置。我们将在这个应用程序中使用两种不同版本的 indexOf 方法。第 153 行所使用的这个方法只接收惟一的一个参数,即所要查找的子字符串。indexOf 方法能够在 htmlText 中定位出所选定的子字符串(本例中,为所选拍卖物品的名称)首次出现时的位置。此方法将返回该子字符串在 htmlText 中首次出现时的起始位置上的索引。如果 htmlText 中并不存在所指定的子字符串,则方法 indexOf 返回 -1。第 153 行是把所得到的结果存储在 itemLocation 中。

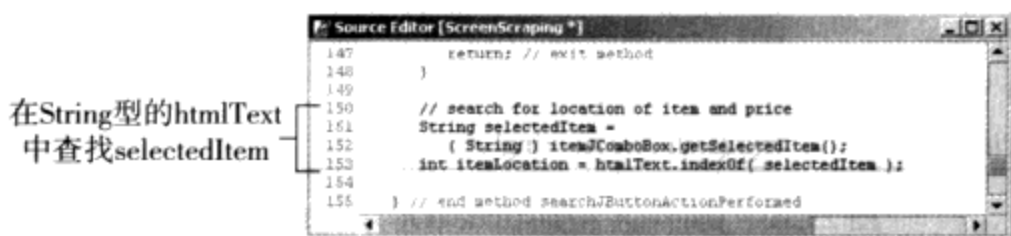


图 23.7 定位所选定的拍卖物品的名称

5. 定位价格的起始位置 将图 23.8 中第 154 行添加到 searchJButtonActionPerformed 方法中,第 154 行用于定位拍卖物品价格起始处的索引值。本代码行使用的是接收两个参数的 indexOf 方法,这两个参数,一个为所要查找的子字符串("€"),另一个则为子字符串查找开始时的起始位置处的索引,此方法将不会查看出现在起始索引(通过 itemLocation 指定)位置前的任何字符。对于本应用程序来说,此时只需要关注含所选拍卖物品价格的那部分 htmlText。我们知道,在所选拍卖物品名称之后出现的价格便是我们需要的价格,该价格应起始于 "€" 子字符串。由 indexOf 方法返回的索引将存储在变量 priceBegin 中。

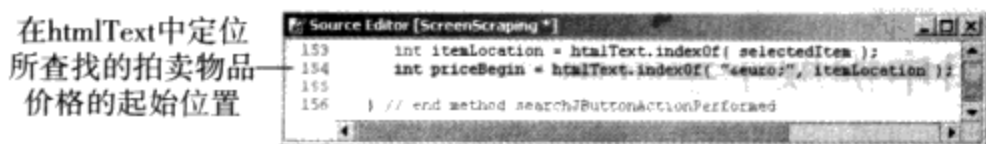


图 23.8 定位所选拍卖物品的价格

6. 定位价格的结束位置 将图 23.9 中第 155 行插入到 searchJButtonActionPerformed 方法中。第 155 行通过向 indexOf 方法传递一个字符串 "</TD>" 及一个起始索引 (priceBegin), 来查找所需价格的索引位置。在 HTML 字符串中每个价格的后面,都直接跟有一个 </TD> 标签(不含空格),因此,priceBegin 之后出现的首个 </TD> 标签的索引将作为当前价格的结束标记。标签 <TD> 和 </TD> 用于显示表中的数据。同

样，这里并不需要理解有关 HTML 表格方面的知识。在本教程中，只需要知道每个价格的后面都有一个能够标记表格内数据结束的</TD>。indexOf 方法所返回的索引将被存储在变量 priceEnd 中。

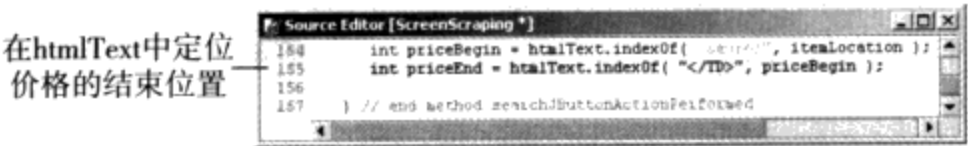


图 23.9 定位所选拍卖物品价格的结束位置

7. 保存应用程序 保存修改后的源代码文件。目前，我们已对所选物品价格起始处的索引（priceBegin）及该价格结束位置之后的那一个索引（priceEnd）进行了保存。但此时仍没有编写用于显示输出的代码，所以如果现在编译并运行该应用程序的话，点击 Search JButton 以后，Price:JTextField 中将不会显示任何价格。此项功能将随后得到添加。

另外，String 类中还有一个与 indexOf 方法非常类似的方法——lastIndexOf 方法。该方法用来定位某字符串中的子字符串最后一次出现时的位置，即从字符串的末尾开始，执行向后查找。这个 lastIndexOf 方法同样也是返回某字符串中所指定子字符串的起始位置处的索引。如果所指定的子字符串并未出现在所查询的字符串中，那么，lastIndexOf 方法将返回 -1。

对于 lastIndexOf 方法，String 中也将使用两种方式查找子字符串。第一种方式接收一个参数，该参数为所要查找的子字符串。第二种方式接收两个参数——一个参数是所要查找的子字符串，另一个参数则为子字符串进行向后查找时的开始位置处的索引。图 23.10 中列出了一些通过调用 indexOf 方法和 lastIndexOf 方法来查找子字符串的例子。

方法	表达式举例（假设 text = "My String is a long String"）	返回值
indexOf(String)	text.indexOf("ring")	5
indexOf(String, int)	text.indexOf("ring", 10)	22
lastIndexOf(String)	text.lastIndexOf("ring")	22
lastIndexOf(String, int)	text.lastIndexOf("ring", 3)	-1

图 23.10 indexOf 和 lastIndexOf 方法的调用举例

自测题

- 通过调用 _____ 方法可在某个字符串中定位出一个子字符串首次出现时的位置。
a) indexOf b) firstIndexOf c) findFirst d) locate
- 在一个接收 String 和 int 作为其参数的 lastIndexOf 方法中，传递至 lastIndexOf 方法中的第 2 个参数表示 _____。
a) 所要查找的字符个数 b) 采用向前查找法开始进行查找时的起始位置处的索引
c) 需要定位的子字符串的长度 d) 采用向后查找法开始进行查找时的起始位置处的索引

答案：1) a 2) d

23.5 从字符串中提取子字符串

下面，将学习如何使用 substring 方法从 HTML 字符串中检索出所选拍卖物品的价格。substring 方法将返回一个新的 String 对象，该对象包含了现有 String 对象中所指定部分的一个字符串拷贝。

检索出所选定拍卖物品的价格

- 提取价格 将图 23.11 中第 157 行至第 160 行添加到方法 searchJButtonActionPerformed 中。第 158 行至第 159 行通过使用 substring 方法从 htmlText 中提取相应的价格。其中，第一个参数（priceBegin + 6）

为一个起始索引，即该方法将从此位置处开始复制原String中的字符。前面曾经讲过，priceBegin中包含的是当前拍卖物品价格开始位置处（含字符串"€"）的索引。由于需要将货币值显示为美元，因而并不需要这个欧元符号；通过给priceBegin加6，可越过前6个字符（即字符串"€"）。第2个参数（priceEnd）所指定的这个索引位置，为最后一个需复制的字符的下一个字符索引（即从起始索引一直复制到第2个参数索引所指定的子字符，但并不包括该子字符）。变量priceEnd中存储了文本"</TD>"开始位置处的索引，它紧接着当前拍卖物品的价格之后。第158行所返回的子字符串（priceText）包含从原String中得到的一个指定部分的拷贝。在本例中，所返回的这个子字符串为该拍卖物品的价格（不含欧元符号）。例如，若所选择的拍卖物品为Antique Rocking Chair，price中将存储字符串"82.67"，这与我们在图23.1中看到的数据是一样的（此值位于HTML中的第2行文本内）。

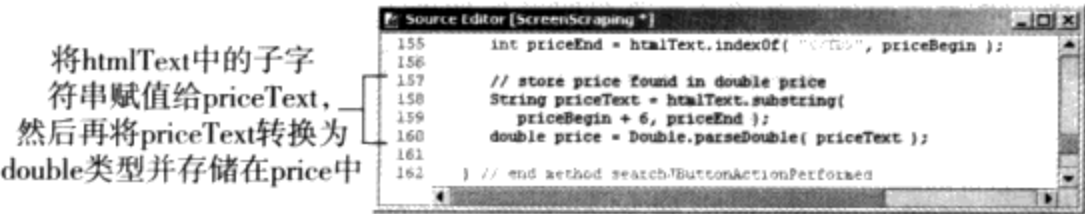


图 23.11 检索出所需要的价格

String类中提供了两种类型的substring方法。其中一种方法是接收一个用来指定某个起始索引的int型参数，即从该位置处开始，复制原String中的字符并最终将这个从起始索引开始直到原字符串结束的子字符串返回。在屏幕抓取应用程序中，将不会使用到这种substring方法。注意，第160行将priceText转换成了一个double类型的值（存储在变量price中），这样便可实现其与转换汇率之间的相乘运算。

- 2. 从欧元转换为美元 为得到拍卖物品的美元价格，必须把HTML中所提取的价格乘以由用户所输入的转换汇率。将图23.12中第162行至第165行添加到searchJButtonActionPerformed方法中。第163行至第160行用于取得转换汇率并将其存储在一个double型的变量conversionRate中。第165行是将该值与price相乘并存储在price中。回顾我们曾经讲过，运算符*=的含义是将其左操作数与右操作数相乘，并将结果存储在左操作数中。

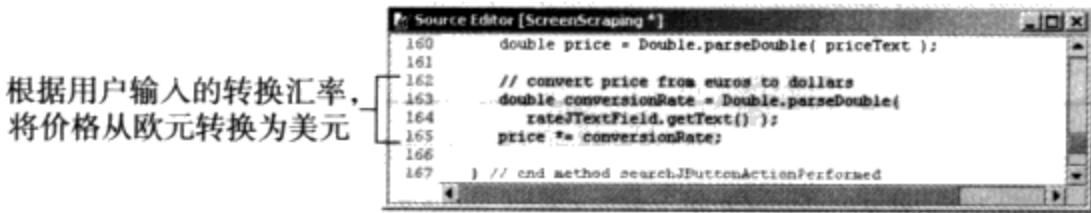


图 23.12 将价格从欧元转换为美元

- 3. 显示价格 将图23.12中第167行至第169行添加到searchJButtonActionPerformed方法中。第169行通过使用第168行上所创建的DecimalFormat对象（dollars），完成了对美元价格的格式化及其显示的处理。

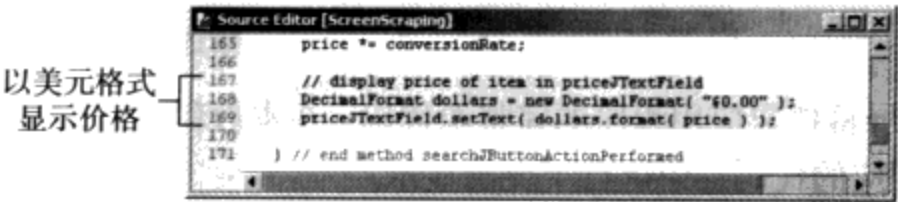


图 23.13 将价格显示在priceJTextField中

- 4. 保存应用程序 保存修改后的源代码文件。
- 5. 打开命令提示符窗口改变目录 选择Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入cd C:\SimplyJava\ScreenScraping进入到当前工作目录中。
- 6. 编译应用程序 键入javac ScreenScraping.java编译该应用程序。
- 7. 运行应用程序 若能正确编译应用程序，键入java ScreenScraping来运行它。图23.14显示了完成后的应用程序的运行结果。从Item:JComboBox中选择一个所要拍卖的物品并输入一个相应的转换汇率，然后点击Search JButton，可以看到，该拍卖物品的价格是以美元价格的形式来显示的。



图 23.14 运行完成后的屏幕抓取应用程序

8. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
9. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

自测题

1. substring 方法 _____。
- a) 可接收一个或两个参数

b) 返回一个新的 String 对象

c) 通过复制现有 String 对象中的一部分从而创建一个新的 String 对象

d) 以上答案都正确
2. 当调用含两个参数的 substring 方法时，第 2 个参数表示 _____。
- a) 通过此处位置上的索引向后进行复制

b) 需复制的子字符串的长度

c) 超过最后一个所要复制的字符的下一个字符的索引

d) 所要复制的最后一个字符的索引

答案：1) d 2) c

23.6 String 类中的其他方法

String 类还为开发人员提供了其他一些操作字符串的方法，图 23.15 中列出了其中一些常用的方法。

方法	描述	表达式举例（假设 text = " My String"）
startsWith(String)	如果某个字符串起始于参数字符串，则返回 true；否则，返回 false	text.startsWith("Your"); 返回: false
endsWith(String)	如果某个字符串终止于参数字符串，则返回 true；否则，返回 false	text.endsWith("ing"); 返回: true
trim()	删除某个字符串中所有空格	text.trim(); 返回: "My String"

图 23.15 一些 String 类中方法的描述

图 23.16 中给出了屏幕抓取应用程序的完整源代码。在本教程中，凡需要添加、查看或者是修改的代码，均在图中相应的代码行中进行了突出显示。

```
1 // Tutorial 23: ScreenScraping.java
2 // Search an HTML code String for an item and display its price
3 // converted from euros to American dollars.
4 import java.awt.*;
5 import java.awt.event.*;
6 import java.text.*;
7 import javax.swing.*;
8
9 public class ScreenScraping extends JFrame
10 {
```

```

11 // JLabel and JComboBox for item names
12 private JLabel itemJLabel;
13 private JComboBox itemJComboBox;
14
15 // JLabel and JTextField for conversion rate
16 private JLabel rateJLabel;
17 private JTextField rateJTextField;
18
19 // JLabel and JTextField for item price
20 private JLabel priceJLabel;
21 private JTextField priceJTextField;
22
23 // JButton to search HTML code for item price
24 private JButton searchJButton;
25
26 // JLabel and JTextArea for HTML code
27 private JLabel sourceJLabel;
28 private JTextArea sourceJTextArea;
29
30 // items that can be found in HTML code
31 private String[] items = { "Antique Rocking Chair" ,
32 "Silver Teapot" , "Gold Pocket Watch" };
33
34 // small piece of HTML code containing items and prices
35 private String htmlText = "<HTML><BODY><TABLE>"
36 + "<TR><TD>Antique Rocking Chair</TD>"
37 + "<TD>&euro;82.67</TD></TR>"
38 + "<TR><TD>Silver Teapot</TD>"
39 + "<TD>&euro;64.55</TD></TR>"
40 + "<TR><TD>Gold Pocket Watch</TD>"
41 + "<TD>&euro;128.83</TD></TR>"
42 + "</TABLE></BODY></HTML>";
43
44 // no-argument constructor
45 public ScreenScraping()
46 {
47     createUserInterface();
48 }
49
50 // create and position GUI components; register event handlers
51 private void createUserInterface()
52 {
53     // get content pane for attaching GUI components
54     Container contentPane = getContentPane();
55
56     // enable explicit positioning of GUI components
57     contentPane.setLayout( null );
58
59     // set up itemJLabel
60     itemJLabel = new JLabel();
61     itemJLabel.setBounds( 8 , 16 , 40 , 21 );
62     itemJLabel.setText( "Item:" );
63     contentPane.add( itemJLabel );
64
65     // set up itemJComboBox
66     itemJComboBox = new JComboBox( items );
67     itemJComboBox.setBounds( 56 , 16 , 184 , 21 );
68     contentPane.add( itemJComboBox );
69
70     // set up rateJLabel
71     rateJLabel = new JLabel();

```

模板中提供的用以描述三件拍卖物品名称和欧元价格的 HTML。


```
72     rateJLabel.setBounds( 8 , 48, 40 , 21 );
73     rateJLabel.setText( "Rate:" );
74     contentPane.add( rateJLabel );
75
76     // set up rateJTextField
77     rateJTextField = new JTextField();
78     rateJTextField.setBounds( 56 , 48, 184, 21 );
79     rateJTextField.setHorizontalAlignment( JTextField.RIGHT );
80     contentPane.add( rateJTextField );
81
82     // set up priceJLabel
83     priceJLabel = new JLabel();
84     priceJLabel.setBounds( 8 , 80 , 40 , 21 );
85     priceJLabel.setText( "Price:" );
86     contentPane.add( priceJLabel );
87
88     // set up priceJTextField
89     priceJTextField = new JTextField();
90     priceJTextField.setBounds( 56 , 80 , 96 , 21 );
91     priceJTextField.setHorizontalAlignment( JTextField.CENTER );
92     priceJTextField.setEditable( false );
93     contentPane.add( priceJTextField );
94
95     // set up searchJButton
96     searchJButton = new JButton();
97     searchJButton.setBounds( 160, 80 , 80 , 23 );
98     searchJButton.setText( "Search" );
99     contentPane.add( searchJButton );
100    searchJButton.addActionListener(
101
102        new ActionListener() // anonymous inner class
103        {
104            // event handler called when searchJButton is pressed
105            public void actionPerformed((ActionEvent event) )
106            {
107                searchJButtonActionPerformed( event );
108            }
109
110        } // end anonymous inner class
111
112    ); // end call to addActionListener
113
114    // set up sourceJLabel
115    sourceJLabel = new JLabel();
116    sourceJLabel.setBounds( 8 , 112 , 48, 16 );
117    sourceJLabel.setText( "Source:" );
118    contentPane.add( sourceJLabel );
119
120    // set up sourceJTextArea
121    sourceJTextArea = new JTextArea();
122    sourceJTextArea.setBounds( 8 , 136, 232, 105 );
123    sourceJTextArea.setText( htmlText );
124    sourceJTextArea.setLineWrap( true );
125    contentPane.add( sourceJTextArea );
126
127    // set properties of application's window
128    setTitle( "Screen Scraping" ); // set title bar string
129    setSize( 259, 278 );           // set window size
130    setVisible( true );           // display window
131
132 } // end method createUserInterface
```

```

133
134 // find and display price substring
135 private void searchJButtonActionPerformed((ActionEvent event) )
136 {
137     // get rate
138     String rate = rateJTextField.getText();
139
140     // rate is an empty string
141     if ( rate.equals( "" ) )
142     {
143         JOptionPane.showMessageDialog( null,
144             "Please enter conversion rate first.",
145             "Missing Rate", JOptionPane.WARNING_MESSAGE );
146
147         return ; // exit method
148     }
149
150     // search for location of item and price
151     String selectedItem = 检索已选取的拍卖物品并定位
152         ( String ) itemJComboBox.getSelectedItemAt(); 其欧元价格的起始位置及
153     int itemLocation = htmlText.indexOf( selectedItem ); 结束位置的索引
154     int priceBegin = htmlText.indexOf( "&euro;" , itemLocation );
155     int priceEnd = htmlText.indexOf( "</TD>" , priceBegin );
156
157     // store price found in double price
158     String priceText = htmlText.substring( 将priceText设置为htmlText中的
159         priceBegin + 6 , priceEnd ); 子字符串, 同时将priceText转换
160     double price = Double.parseDouble( priceText ); 为double型数据并存储在price中
161
162     // convert price from euros to dollars
163     double conversionRate = Double.parseDouble(
164         rateJTextField.getText() ); 利用输入的转换汇率将
165     price *= conversionRate; 价格从欧元转换为美元
166
167     // display price of item in priceJTextField
168     DecimalFormat dollars = new DecimalFormat( "$0.00" );
169     priceJTextField.setText( dollars.format( price ) ); 以美元格式显示价格
170
171 } // end method searchJButtonActionPerformed
172
173 // main method
174 public static void main( String args[] )
175 {
176     ScreenScraping application = new ScreenScraping();
177     application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
178
179 } // end method main
180
181 } // end class ScreenScraping

```

图 23.16 屏幕抓取应用程序的源代码

自测题

1. _____ 方法可删除某个字符串中的所有空格。
a) removeSpaces b) squeeze c) trim d) truncate
2. 如果某个字符串起始于参数字符串, 那么调用 startsWith 方法时将返回 _____。
a) true b) false c) 1 d) 子字符串的索引

答案: 1) c 2) a

23.7 小结

本教程向读者介绍了如何使用String类的方法来处理与字符串相关的操作,以及如何创建和处理String对象。通过学习String类的indexOf和substring方法,我们知道了如何定位、检索以及替换某个字符串中的子字符串。接着,运用Java字符串的知识创建一个简单的屏幕抓取应用程序。了解到屏幕抓取技术是实现从Web页面中的HTML内提取出所需要信息的一个过程。还了解了String类中其他一些如length, charAt, endsWith, startsWith, trim方法的使用,通过它们进一步加强了对字符串访问及其处理的学习。

在下一个教程中,将学习如何在自己的应用程序中使用异常处理机制。所谓异常处理是指对应用程序中出现的问题执行相应解决方案的一项技术。利用异常处理可以让自己的应用程序在不终止执行的情况下,能够检测并处理所出现的问题。我们将改进购车还贷计算器应用程序,使其利用异常处理技术识别并响应用户所输入的无效数据。

技术小结

确定字符串的长度

- 使用String类的length方法

在字符串中定位子字符串

- 利用String类的indexOf方法在字符串中定位某个子字符串首次出现的位置,其返回值为该子字符串首次出现时其起始位置上的索引。如果该子字符串未能在原字符串中找到,则返回-1。
- 利用String类的lastIndexOf方法在字符串中定位某个子字符串最后一次出现的位置,其返回值为该子字符串最后一次出现时其起始位置上的索引。如果该子字符串未能在原字符串中找到,则返回-1。

从字符串中取出子字符串

- 通过String类中含一个参数的substring方法来取得一个子字符串,该子字符串包括从给定起始索引(参数)处开始直到原字符串结束的所有字符。
- 利用String类中含两个参数的substring方法来取得一个子字符串,该子字符串包括从给定起始索引(第一个参数)处开始一直到结束索引(第二个参数)之前的所有字符。

确定字符串的起始子字符串和结束子字符串

- 利用String类的startsWith方法判定某个字符串是否起始于一个给定的子字符串。
- 利用String类的endsWith方法判定某个字符串是否结束于一个给定的子字符串。

去除字符串中的空格

- 利用String类的trim方法去除字符串中从开始字符到结束字符之间的所有空格。

关键术语

String类的charAt方法 返回某个字符串中一个指定位置上的字符。

String类的endsWith方法 若某个字符串结束于一个给定的子字符串,则返回true;否则,返回false。

HTML (HyperText Markup Language, 超文本标记语言) 一项描述Web内容及其格式的技术。

不可变的 Java中的字符串属于不可变的,即字符串在创建以后其内容将不可更改。

String类的indexOf方法 返回某字符串中一个子字符串首次出现时的索引;若该子字符串没有找到,则返回-1。

String类的lastIndexOf方法 返回某字符串中一个子字符串最后一次出现时的索引;若该子字符串没有找到,则返回-1。

String 类的 length 方法 返回某个字符串的字符个数。

文本型 String 对象 通过将一个字符序列写在一对双引号中而形成的字符串常量 (也称字符串常量或字符串文本)。

屏幕抓取技术 从构成 Web 页面的 HTML 中提取出所需要信息的一种过程。

特殊字符 一些如 +, -, *, /, \$, 空格等字符。特殊字符不包括字母和数字。

String 类的 startsWith 方法 若某个字符串起始于一个给定的子字符串, 则返回 true; 否则, 返回 false。

String 一个可将所表示的字符序列当做一个单独单元来看待的类。

字符串常量 字符串文本或文本型 String 对象的另一种称呼。

字符串文本 字符串常量或文本型 String 对象的另一种称呼。

子字符串 由某个字符串中的一部分或其全部所构成的一个字符序列。

String 类的 substring 方法 从现有的 String 对象中复制一个指定的部分从而返回一个新的 String 对象的方法。

String 类的 trim 方法 删除某字符串中从起始位置到结束位置之间的所有空格, 从而返回一个新的 String 对象的方法。

Java 类库索引

String 该类将一个字符序列当做一个独立单元来看待。

● 方法

charAt 返回字符串中某个指定位置上的字符。

endsWith 若某个字符串结束于一个给定的子字符串, 返回 true; 否则, 返回 false。

equals 比较两个字符串中的内容, 假若这两个字符串均具有相同次序且包含相同字符, 则返回 true; 否则, 返回 false。

indexOf 返回某字符串中一个子字符串首次出现时的索引。若未找到该字符串, 则返回 -1。

lastIndexOf 返回某字符串中一个子字符串最后一次出现时的索引。若未找到该字符串, 则返回 -1。

length 返回某个字符串的字符个数。

startsWith 若某个字符串起始于一个给定的子字符串, 则返回 true; 否则, 返回 false。

substring 从现有 String 对象中复制一个指定部分, 以返回一个新的 String 对象。

trim 删除某个字符串中从起始位置到结束位置之间的所有空格, 返回一个新的 String 对象。

valueOf 能够将数字值转换为一个字符串。

习题

选择题

- 23.1 从 Web 页面中提取所需要信息的过程被称为 _____。
- a) Web 漫步 b) 屏幕抓取 c) 查询 d) 重定位
- 23.2 假若 indexOf 方法未能找到所指定的子字符串, 则返回 _____。
- a) false b) 0 c) -1 d) 以上答案都不对
- 23.3 String 类允许开发人员 _____。
- a) 查找字符串 b) 从字符串中检索字符
- c) 确定字符串中的字符个数 d) 以上答案都正确
- 23.4 _____ 是一项描述 Web 内容的技术。
- a) String 类 b) 字符串文本 c) HTML d) 屏幕抓取器
- 23.5 _____ 方法用于返回字符串中一个指定索引上的字符。
- a) get b) char c) getAt d) charAt
- 23.6 _____ 方法通过复制现有 String 对象中的一部分, 以创建一个新的 String 对象。
- a) stringCopy b) substring c) copyString d) copySubString

- h) 编译应用程序 键入 `javac SupplyCalculator.java` 对应用程序进行编译。
- i) 运行完成后的应用程序 若能正确编译应用程序, 键入 `java SupplyCalculator` 来运行它。为测试应用程序, 从列表中添加一些商品而后再删除一些商品, 最后查看一下总价格的计算是否正确。
- j) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- k) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

23.12 (加密应用程序) 编写一个能够将用户输入的信息进行加密的应用程序 (如图 23.18 所示)。该应用程序可按照两种方式进行加密: 置换加密和调换加密 (后面会有相应的描述)。用户通过一个 `TextField` 输入信息, 然后选择所需要的加密方式, 加密后的信息则立刻显示在一个不可编辑的 `TextField` 中。

在置换加密算法中, 英文字母表中的每一个字符都将被置换字符串中一个不同的字符所替换。在对应到英文句子中, 每一个出现的字母都将被置换字符串中相应索引上的字符所替换。例如, 要对一个字符串 "code" 采取置换加密, 如果字符 "c", "o", "d", "e" 分别对应置换字母表中的 "e", "f", "g", "h", 那么, 经加密后的原字符串将变为 "efgh"。

在调换加密算法中, 需要创建两个字符串。第一个字符串包含所有位于输入字符串偶数索引位置上的字符, 第二个字符串则包含所有奇数索引位置上的字符。将第二个字符串追加到第一个字符串之后, 它们之间还需放置一个空格字符。例如, 采用调换加密, 单词 "code" 将变为 "cd oe"。

- a) 将模板复制到工作目录中 将 `C:\Examples\Tutorial23\Exercises\CipherEncryption` 目录复制到 `C:\SimplyJava` 目录中。
- b) 打开模板文件 在自己的文本编辑器中打开 `Encryption.java` 文件。

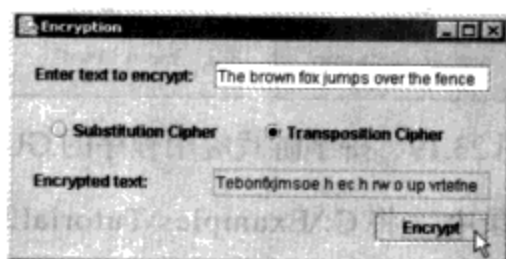


图 23.18 加密应用程序的 GUI

- c) 向 `substitutionCipher` 方法中添加代码 在 `substitutionCipher` 方法中 (第 126 行至第 140 行), 已分别声明了用于记录英文字母表和置换字母表的两个字符串, 它们是 `normalAlphabet` 和 `cipherAlphabet`。需要使用的其他字符串也已为读者提供——`cipher` 是一个用于存储密文的空字符串, `plain` 中将存储用户输入的文本。在这些字符串声明语句的后面, 添加一条空的 `for` 语句, 用于循环 `plain` 中所存储的每一个字符。
- d) 完成置换加密算法 在步骤(c)中所添加的 `for` 语句的内部, 创建一个 `int` 型变量 `index` 并为其赋予当前 `plain` 中的字符在 `normalAlphabet` 中所对应的索引值。若 `index` 不等于 -1, 则取出 `cipherAlphabet` 中该 `index` 上所存储的字符并将其追加到 `cipher` 的末尾。此时, 原字符串将被相应的密码字符所取代。在该 `for` 语句的后面, 添加用于在 `cipherJTextField` 中显示 `cipher` 的代码。
- e) 向 `transpositionCipher` 方法中添加代码 找到 `transpositionCipher` 方法, 该方法紧随 `substitutionCipher` 方法之后。在这一方法中, 字符串 `firstWord`, `lastWord` 以及 `plainText` 已为读者声明, 其中, `plainText` 用于存储用户输入的文本。在这些字符串声明语句的后面, 添加一条空的 `for` 语句, 用于循环 `plainText` 中所存储的每一个字符。
- f) 完成调换加密算法 在步骤(e)中所添加的 `for` 语句内部, 再添加一条 `if...else` 语句, 并当计数器变为偶数时, 执行其中的 `if` 语句 (提示: 偶数除以 2, 所得的余数为 0)。在 `if` 语句的内部, 取出 `plainText` 中的当前字符并将其追加到 `firstWord` 的末尾。在 `else` 语句的内部, 同样也是取出 `plainText` 中的当前字符并将其追加到 `lastWord` 的末尾。在该 `for` 语句的后面, 添加相应的代码, 实现在 `cipherJTextField` 中显示由 `firstWord`, 空格, `lastWord` 所组成的字符串。

- g) 保存应用程序 保存修改后的源代码文件。
- h) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\Encryption` 进入到当前工作目录中。
- i) 编译应用程序 键入 `javac Encryption.java` 编译该应用程序。
- j) 运行完成后的应用程序 若能正确编译应用程序, 输入 `java Encryption` 来运行它。通过输入不同的文本并使用这两种加密方式对该应用程序的测试, 检验所得的结果是否与图 23.18 中所示的结果相类似。
- k) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- l) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

23.13 (拼字游戏应用程序) 编写一个拼字游戏应用程序, 该应用程序中有一个数组, 其中存储了一些预先定义好的单词。此游戏会随机挑选出一个单词并打乱其中的字母顺序 (如图 23.19 所示)。具体方法是: 取出该单词中的第一个字母然后再将其放回到原字符串中的一个随机位置上, 这样重复 20 次以便完全打乱原字符串。为方便用户进行猜测, 该应用程序提供了一个用于显示被打乱单词的 JLabel。如果用户猜测正确, 则显示一条信息并使用一个不同的单词来继续这一过程。如果用户猜测错误, 同样也显示一条信息并允许用户再次进行尝试。

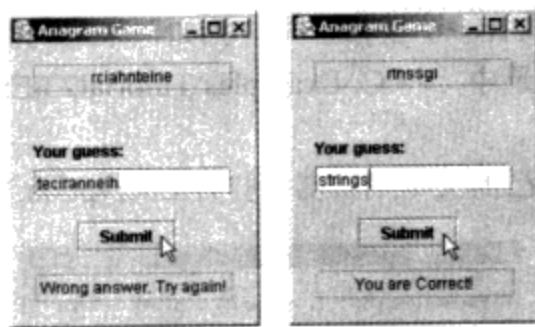


图 23.19 拼字游戏应用程序的 GUI

- a) 将模板复制到工作目录中 将 `C:\Examples\Tutorial23\Exercises\Anagram` 目录复制到 `C:\SimplyJava` 目录中。
- b) 打开模板文件 在自己的文本编辑器中打开 `Anagram.java` 文件。
- c) 在 `generateAnagram` 方法中添加一条 `for` 语句 `generateAnagram` 方法 (位于第 133 行至第 145 行) 用于从一个预先定义好的数组中选择一个字符串并打乱其中的字符排列。打乱后的字符串被存储在变量 `scrambled` 中, 同时由该字符串所生成的一个随机索引也将相应地被存储在变量 `randomIndex` 中。只需在本练习中完成这个 `generateAnagram` 方法 (其余部分已为读者提供)。在 `generateAnagram` 方法内部位于变量声明语句的后面, 添加一条可循环 20 次的空白 `for` 语句。
- d) 生成打乱的单词 在步骤(c)中所添加的 `for` 语句的内部, 声明一个 `char` 型变量 `firstCharacter`, 并为其赋予 `scrambled` 中的第一个字符。然后, 通过使用 `substring` 方法删除 `scrambled` 中的第一个字符。之后, 创建 `String` 型变量 `temporary1` 和 `temporary2`。在 `temporary1` 中, 存储 `scrambled` 内从起始位置到 `randomIndex` 之间的所有字符。在 `temporary2` 中, 存储 `scrambled` 内剩余的字符。然后, 把 `firstCharacter` 追加到 `temporary1` 的末尾, 最后应实现 `temporary1` 和 `temporary2` 的连接并将其结果存储到 `scrambled` 中。此时, 我们已将 `scrambled` 中的第一个字符移动到了一个随机位置上。这一过程需重复 20 次以便进一步打乱字符串中的字符排列, 并最终为用户显示出这一结果。
- e) 生成一个随机索引 为了让 `for` 语句进行下一轮迭代, 还需生成一个新的 `randomIndex`。在位于步骤(d)中连接 `temporary1` 和 `temporary2` 语句的后面, 使用 `randomGenerator` (一个已在模板中创建的 `Random` 对象) 生成一个随机整数, 范围从 0 至 `scrambled` 中最后一个索引值 (含边界值), 同时将所生成的这个随机数存储到变量 `randomIndex` 中。
- f) 显示拼字 在步骤(c)至步骤(e)中所完成的 `for` 语句的后面, 添加可将 `scrambled` 显示到 `anagramJTextField` 中的语句。

- g) 保存应用程序 保存修改后的源代码文件。
- h) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\Anagram` 进入到当前工作目录中。
- i) 编译应用程序 键入 `javac Anagram.java` 编译该应用程序。
- j) 运行完成后的应用程序 若能正确编译应用程序, 键入 `java Anagram` 来运行它。通过输入不同单词正确和错误的猜测结果, 完成对应用程序的测试。
- k) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- l) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

23.14 (说出这段代码的作用) 下面这段代码在执行完以后, 数组 `words` 中所存储的字符串将是什么?

```
1 String words[] = { "dance", "walk", "talking", "eat" };
2
3 for ( int counter = 0 ; counter <= words.length - 1 ; counter++ )
4 {
5     if ( words[ counter ].endsWith( "e" ) )
6     {
7         words[ counter ] = words[ counter ].substring(
8             0 , words[ counter ].length() - 1 );
9     }
10
11     if ( !( words[ counter ].endsWith( "ing" ) ) )
12     {
13         words[ counter ] += "ing";
14     }
15
16 } // end for loop
```

23.15 (找出代码中的错误) 下面这段代码将删除 `test` 中的所有空格。请找出代码中的错误。

```
1 String test = "s p' a c e s" ;
2 int index;
3
4 while( test.indexOf( " " ) == -1 )
5 {
6     index = test.indexOf( " " );
7     test = test.substring( 0 , index - 1 ) + test.substring( index );
8 }
```

挑战题

23.16 (“儿童黑话”应用程序) 编写一个可将英文句子编码为“儿童黑话”的应用程序(如图 23.20 所示)。所谓“儿童黑话”其实是一种经过编码的语言形式, 通常用于娱乐。通过以下算法来创建“儿童黑话”中所使用的单词:

为了从英文句子中创建“儿童黑话”, 需对每一个单词进行翻译。将英文单词译为“儿童黑话”中所使用的单词, 其规则是: 把每一个英文单词的首字母(非元音字母)放置在该英文单词的末尾并添加两个字母“ay”。如果该英文单词的首字母为一个元音字母, 则将其放置在该单词末尾的同时还需添加一个字母“y”。采用这种方法, 单词“jump”变为“umpjay”, 单词“the”变为“hetay”, 而单词“ace”则变为“ceay”。另外, 单词之间的空格仍然保持不变。

我们做下列假设: 用户输入的英文句子是由空格所分隔的一些单词组成的, 它们之间没有标点符号, 且所有单词都至少由两个以上的字母组成。使用 `translateToPigLatin` 方法最终将把英文句子逐字翻译为“儿童黑话”。

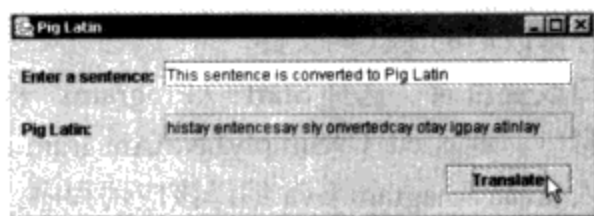


图 23.20 “儿童黑话”应用程序的 GUI

- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial23\Exercises\PigLatin 目录复制到 C:\SimplyJava 目录中。
- b) 打开模板文件 在自己的文本编辑器中打开 PigLatin.java 文件。
- c) 在 translateToPigLatin 方法中添加一条 for 语句 在 translateToPigLatin 方法中 (位于第 97 行至第 109 行), 已经声明了四个变量——String 型变量 prefix 和 suffix (利用其完成单词的转换)、String 型数组 words (包含用户输入的文本中的所有单词)、String 型的 translatedText (存储经翻译以后所得到的句子)。在变量声明语句的后面, 添加一条 for 语句, 循环 words 内的每一个单词。
- d) 获取单词中的首字母 在步骤(c)中所添加的 for 语句的内部, 取出当前单词中的首字母并将该字母的小写形式存储到 prefix 中 (提示: 使用 substring 方法来取得第一个字母并将其赋予 prefix。然后, 通过 prefix 调用 String 的 toLowerCase 方法以返回相应字母的小写形式)。
- e) 确定后缀 在取出第一个字母之后, 添加一条 if...else 语句, 该 if 语句将在 prefix 中的字母为元音字母时 (即当 prefix 中的字母为下列字母之一时: a, e, i, o, u) 开始执行。在 if 语句体内, 将字符串 "y" 赋予 suffix, 在 else 语句中, 将字符串 "ay" 赋予 suffix。
- f) 将当前英文单词转换为“儿童黑话”中使用的单词 在 if...else 语句的后面, 按照下列规则将当前英文单词转换为“儿童黑话”中所使用的单词: 除原单词第一个字母外, 提取所有剩余字母并为这部分追加 prefix 及 suffix 的值, 将得到的新单词存储在数组 words 中的当前位置上。完成以后, 再把这个 words 中的当前元素追加到 translatedText 的末尾, 注意后面还应跟有一个空格。至此, 所有已转换为“儿童黑话”的单词便连接成了一个句子。
- g) 返回新得到的句子 在 for 语句的后面, 修改 return 语句使之返回 translatedText 中的结果。
- h) 保存应用程序 保存修改后的源代码文件。
- i) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\PigLatin 进入到当前工作目录中。
- j) 编译应用程序 键入 javac PigLatin.java 编译该应用程序。
- k) 运行完成后的应用程序 若能正确编译应用程序, 输入 java PigLatin 来运行它。为测试该应用程序, 输入多个不同的句子并将其翻译为“儿童黑话”, 检验每次翻译的结果是否正确。
- l) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- m) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。



教程 24 改进的购车还贷计算器应用程序

介绍异常处理技术

教学目标

在本教程中，读者将学到以下内容：

- 异常处理技术
- 处理异常的 try, catch, finally 及 throws 语句块
- 异常的继承结构
- 检查型异常和非检查型异常

在本教程中，将学习有关异常处理的知识。所谓异常，是指应用程序执行过程中所产生问题的一种表示。“异常”这个名字来源于这样一个事实：尽管可能会出现问题，但问题却不经常发生。如果遵守“规则”是指某条语句在正常情况能够正确执行，那么“异常”表示出现了一个问题。异常处理技术让开发人员通过创建相应的程序解决（或称处理）应用程序执行过程中所出现的异常。通常情况下，针对异常的处理并不影响应用程序的执行，就好像未曾出现过任何问题一样。

本教程将以改进的购车还贷计算器应用程序的探试作为起点，之后，介绍异常处理的概念并演示一些基本的异常处理技术。我们将具体学习如何处理异常，包括 try, catch, finally 及 throws 语句块的使用，还有异常类的继承结构等内容。

24.1 探试改进的购车还贷计算器应用程序

在本教程中，我们将改进教程 8 中的购车还贷计算器应用程序，为其添加异常处理语句。改进后的应用程序必须满足下列需求：

应用程序需求分析

某银行希望通过开发的应用程序阻止用户输入不正确的汽车贷款额。我们在教程 8 中开发的这一应用程序，虽然在输入错误的数据时能够继续运行，但并不能计算出正确的结果。改进的购车还贷计算器应用程序只允许用户在 Price:JTextField 和 Down payment:JTextField 中输入整数，而如果用户输入任何非整数数据（如 double 或其他非数值数据）时，则会显示一个提示用户输入整数数据的消息对话框。同样，用户只能在 Annual interest rate:JTextField 中输入 double 型数据。如果用户输入任何非 double 型数据，则会显示一个提示用户输入一个 0.0~100.0 范围内的 double 值的消息对话框。另外，用户还需要输入利率，例如输入利率 5，等价于 5%。

用户在提供有效输入的前提下，改进的购车还贷计算器应用程序将计算出汽车贷款期限分别为 24, 36, 48 和 60 个月的月度支付额。用户需要输入汽车价格、首期付款额以及相应的年利率大小。我们将以这个完成后的应用程序的探试作为起点。之后，学习一些 Java 技术并最终创建一个属于自己的应用程序。



探试改进的购车还贷计算器应用程序

1. 定位到完成后的应用程序 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\Examples\Tutorial24\CompletedApplication\EnhancedCarPayment` 将目录改变到这个完成后的（改进的）购车还贷计算器应用程序的目录下面。
2. 运行改进的购车还贷计算器应用程序 在命令提示符窗口下键入 `java CarPayment` 运行该应用程序（如图 24.1 所示）。
3. 在 Down payment:JTextField 中输入一个 double 值 在 Price:JTextField 中输入 16900，在 Down payment:JTextField 中输入 6000.50。此时，应用程序的运行窗口如图 24.2 所示。



图 24.1 运行完成后的购车还贷计算器应用程序



图 24.2 在 Down payment:JTextField 中输入一个 double 值

4. 计算月度支付额 通过点击 Calculate JButton 在 JTextArea 中显示月度支付额。注意，此时却出现了一个错误消息对话框（如图 24.3 所示）。

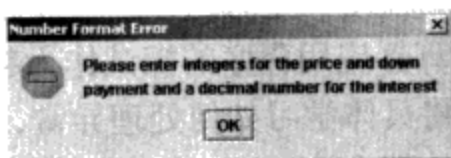


图 24.3 因输入错误的首期付款额而弹出的一个消息对话框

5. 在 Down payment:JTextField 中输入非数字数据 将 Down payment:JTextField 中的输入值 6000.50 改为 600p。应用程序的运行窗口如图 24.4 所示。点击 Calculate JButton，尝试再次在 JTextArea 中显示月度支付额。此时，再次弹出如图 24.3 所示的错误消息对话框（因为需要输入整数值，所以不能有像字符 p 这样的非数字字符）。
6. 在 Annual interest rate: JTextField 中输入非数字数据 将 Down payment:JTextField 中的 600p 更改为 6000。在 Annual interest rate:JTextField 中输入 7.5%。此时，应用程序的窗口如图 24.5 所示。点击 Calculate JButton，尝试在 JTextArea 中显示月度支付额。如图 24.3 所示错误消息对话框将再一次出现（应正确输入值 7.5，输入特殊字符 % 会出现错误）。

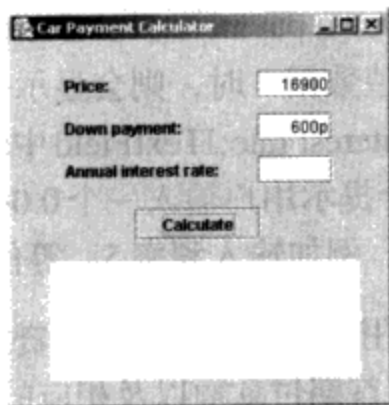


图 24.4 在 Down payment:JTextField 中输入非数字字符

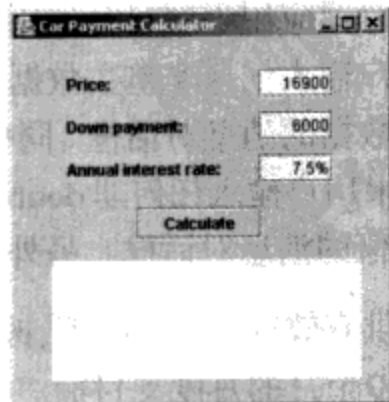


图 24.5 在 Annual interest rate:JTextField 中输入非数字字符

7. 改正输入 将 Annual interest rate:JTextField 中的 7.5% 改为 7.5，通过点击 Calculate JButton 显示计算出的正确月度支付额（如图 24.6 所示）。

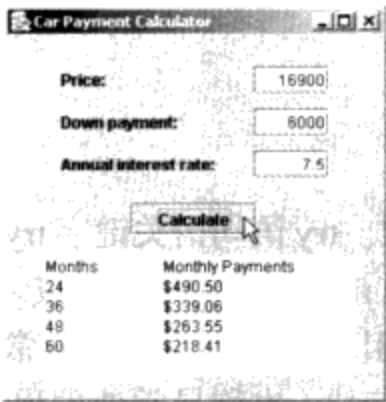


图 24.6 输入改正以后所显示的月度支付额

- 8. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- 9. 闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

24.2 介绍异常处理技术

应用程序的运行逻辑会不断检测该应用程序是否应继续执行。考察下列伪代码：

```
执行一项任务
如果先前任务未能正确执行
    执行错误处理
执行下一个任务
如果先前任务未能正确执行
    执行错误处理
.....
```

在这段伪代码中，执行起始于某项任务。然后便测试该任务是否能正确执行。如果执行有误，则完成一个错误处理。否则，继续执行下一个任务。虽然这一错误检查方式能够起作用，但是，这种将应用程序的逻辑处理与错误处理混杂在一起的做法，会使应用程序难以阅读、修改、维护以及调试（尤其不利于大型应用程序）。事实上，如果一些潜在错误很少发生，那么应用逻辑与错误处理逻辑混杂在一起会大大降低程序执行的效率，这是因为应用程序在每一项任务执行之后必须明确查明是否有错，以此确定下一项任务是否继续执行。

异常处理机制允许开发人员将错误处理代码从应用程序的逻辑代码中分离出来，从而改善应用程序的清晰度和可修改性。开发人员只需处理所关心的异常，包括选择要处理的异常类型或者选择一组相关异常（这些异常隶属同一个继承结构）。通过这种灵活性，极大地降低了错误被忽略的可能性，使应用程序更加健壮。

一个方法在执行的过程中如果出现问题，并且该方法未能对问题进行改正，那么，将抛出一个异常。此时，并不能保证有一个处理此异常的异常处理程序（当应用程序检测到某个异常时所执行的代码）。如果存在这样的异常处理程序，它将捕获并处理该异常，而如果出现未捕获型异常（未能匹配异常处理程序的异常），则很可能会导致应用程序终止。

自测题

- 1. 当应用程序检测到异常时，会调用_____。
a) 异常代码 b) 异常处理器 c) 异常处理程序 d) 以上答案都不对

2. 如果方法未能改正所出现的问题, 则方法将 _____ 异常。

- a) 抛出 b) 捕获 c) 返回 d) 以上答案都不对

答案: 1) c 2) a

24.3 Java 中的异常处理

Java 通过提供 try 语句来处理异常。try 语句由关键字 try, 划分 try 语句块的大括号 ({}), 一个或多个 catch 语句块、一个可选用的 finally 语句块所组成。

使用 try 语句块的目的是将可能引起异常的语句和异常产生时不需要执行的语句包装起来。try 语句块的后面至少有一个 catch 语句块 (也称异常处理程序) 或一个 finally 语句块。每个 catch 语句块需要在圆括号内指定一个参数 (称为异常参数), 该参数表示此异常处理程序所能处理的异常类型。通过参数可以让 catch 语句块与被捕获的异常对象之间进行交互。在最后一个 catch 语句块的后面, 还可以有一个 finally 语句块, 它表示无论是否有异常发生, 都需要执行其代码。在教程 25 中, 将看到 finally 语句块将作为释放资源, 从而防止“资源泄漏”的代码的一个非常理想的位置。

如果 try 语句块中有异常产生, 则 try 语句块会立刻终止执行。与其他代码块一样, 当 try 语句块终止执行时, 语句块中所声明的局部变量都将失去意义。紧接着, 应用程序将查找用于处理此异常类型的第一个 catch 语句块。应用程序通过将抛出的异常类型同每一个 catch 语句块中的参数类型相比较, 从而定位一个可以匹配的 catch 语句块。如果存在类型相同或者被抛出的异常类型为某个异常参数类型的子类, 则表示匹配已找到。如果找到匹配, 那么与之匹配的 catch 语句块将开始执行。当 catch 语句块执行结束时, 其内所声明的局部变量 (以及异常参数) 都将失去作用。与此同时, 其他 catch 语句块也都将被跳过, 并且如果不存在 finally 语句块, 那么程序会转到这些语句块之后的第一行代码上, 否则, 执行 finally 语句块。

总之, 如果不存在同 try 语句块抛出的异常相匹配的 catch 语句块, 同时, 如果也没有 finally 块, 则执行所有 catch 语句块之后的第一行代码, 否则, 执行 finally 语句块。

如果 try 语句块中没有异常产生, 应用程序会忽略同该语句块相对应的所有 catch 语句块, 并且如果不存在 finally 语句块, 则应用程序会执行 catch 语句块之后的下一条语句, 否则, 执行 finally 语句块。无论异常是否在 try 语句块中抛出, 或者是在任何一个 catch 语句块中抛出的, finally 语句块都将执行。

在方法声明的过程中, throws 语句将用来指定该方法所抛出的异常类型。throws 语句位于方法体之前, 参数列表之后。该语句包含一个由逗号分隔的异常列表, 表示方法执行过程中若出现问题时可能抛出的所有异常。异常可以由方法体内语句抛出的, 或者是通过调用其他方法时抛出的。在方法声明过程中可以抛出指定的异常类, 或者是抛出这些异常类的子类。有关 throws 语句的使用不属于本书讨论的范围。

大部分读者在阅读本书的过程中, 需要关心的只是类库中方法所抛出的异常。Java 允许程序员创建属于自己的异常类型并抛出这些异常。为处理程序员定义的异常, 需要在代码中包含一个 throws 语句, 表示可能产生的这些异常, 同时还应包含用于处理这些错误的 catch 语句块。

使用 catch 语句块将确定是否处理异常或者部分处理异常。也就是说, 异常处理程序可将处理 (或者部分处理) 推迟到另外一个 catch 语句块中。至于其他情况, 异常处理程序可通过下面一条 throw 语句重抛异常:

```
throw exceptionReference;
```

其中，exceptionReference 是 catch 语句块中所抛异常的异常参数。当重抛发生时，下一个 try 语句块（如果有的话）会检测重新抛出的异常，并且其中一个 catch 语句块将试图对该异常进行处理。

自测题

- 1. 不论异常是否产生，_____（如果存在）总会执行。
a) catch 语句块 b) finally 语句块 c) catch 和 finally 语句块 d) 以上答案都不对
- 2. 如果 try 语句块中未产生异常，应用程序会忽略相应的_____。
a) finally 语句块 b) return 语句 c) catch 语句块 d) 以上答案都不对

答案：1) b 2) c

24.4 Java 中异常的结构

本节考察 Java 异常的继承结构。注意，异常也是对象，程序员因此也能创建相应的异常类结构。图 24.7 显示了 Throwable 类继承结构中的一部分，Throwable 类是所有异常类的超类，而只有 Throwable 对象可以使用异常处理机制。Throwable 类有两个子类：Exception 类和 Error 类。Exception 类及其子类所表示的异常是 Java 应用程序应该被捕获的异常，而 Error 类及其子类所表示的异常则通常发生在 Java 的运行系统中，并且不必被捕获。

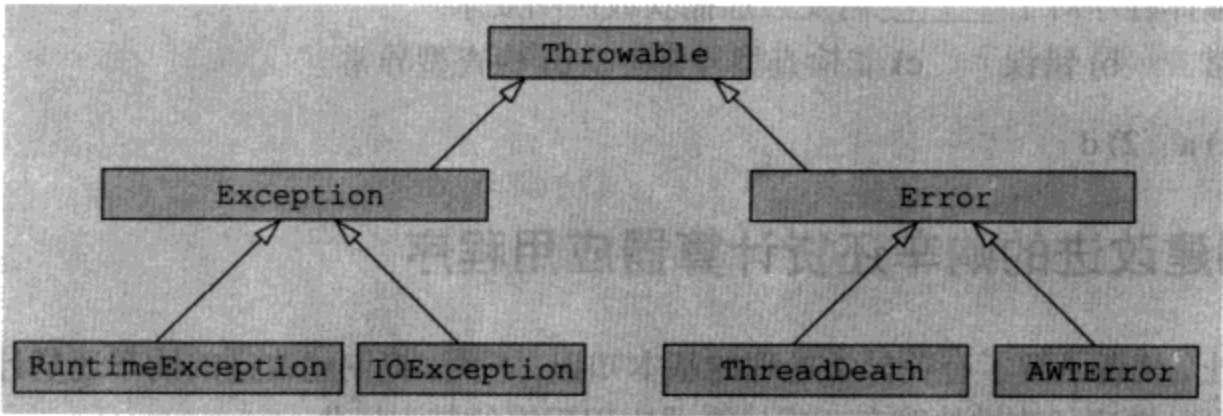


图 24.7 Throwable 类的继承结构

Java 区分两种类型的异常：检查型异常和非检查型异常。RuntimeException 类（位于 java.lang 包中）是所有应用程序运行过程中所抛异常的超类，因此 RuntimeException 类型的子类异常都是非检查型异常，而所有非 RuntimeException 类及其子类的异常则都属于检查型异常。

检查型异常和非检查型异常之间的区别是很关键的，因为 Java 编译程序针对检查型异常强加了一个捕获或声明要求。编译程序通过检查每一个方法调用和方法声明确定方法是否抛出检查型异常。如果抛出检查型异常，编译程序需要保证检查型异常应位于 catch 语句块中（被捕获），或者通过 throws 语句得到声明。为实现捕获，产生异常的代码必须放置在 try 语句块中，并且为该检查型异常提供一个 catch 语句块（或者使用被捕获异常的超类来进行捕获）。为了实现异常的声明，在产生异常的方法参数列表之后、方法体之前提供一个含检查型异常的 throws 语句。如果未能提供捕获或声明，则编译程序将报错并指明该异常应被捕获或声明。



常见编程错误

如果未捕获的异常是一个检查型异常，那么当应用程序编译时会产生错误。



常见编程错误

如果未捕获的异常是一个非检查型异常，那么当应用程序运行时会产生错误。

尽管Java编译程序并不要求提供非检查型异常的异常处理程序,但是对可能产生的异常提供适当的异常处理则是一个好习惯。例如,虽然 `NumberFormatException` 是一个非检查型异常(`NumberFormatException`是`RuntimeException`类的一个子类),但是所有应用程序都应处理由`Integer`类的`parseInt`方法所产生的`NumberFormatException`异常。读者将在随后`NumberFormatException`类的学习过程中了解如何对它进行捕获。

同一个超类可导出多个不同的异常类。如果编写的异常处理程序能够捕获超类的异常对象,那么它也能捕获其子类的所有对象。例如,捕获`RuntimeException`的异常处理程序同样也能够捕获`NumberFormatException`异常。利用这种方式,一个`catch`语句块可以使用更为简洁的形式处理多个相关错误。

如果需要对多个异常进行不同的处理,可以分别捕获各自的子类异常。要想在一个`catch`语句块中捕获多个相关异常,只有当所有子类的异常处理操作都相同时才有意义。否则,就应该为每一个子类异常使用独立的`catch`语句块来进行单独处理。对于这种情况,必须将子类异常的处理放置在超类异常处理的前面。

自测题

1. 所有属于 _____ 子类的异常都是非检查型异常。
a) `RuntimeException` b) `Exception` c) `Error` d) 以上答案都不对
2. Java 编译程序对于 _____ 需要强加捕获或声明要求。
a) 异常 b) 错误 c) 非检查型异常 d) 检查型异常

答案: 1) a 2) d

24.5 创建改进的购车还贷计算器应用程序

目前为止,读者了解了有关异常处理的基本知识,下面,将创建改进的购车还贷计算器应用程序。以下伪代码描述了改进的购车还贷计算器应用程序的基本操作:

当用户点击 `Calculate` `JButton` 时

清除原先 `JTextArea` 中的文本

try

从 `Price:JTextField` 中获取汽车价格

从 `Down Payment:JTextField` 中获取首期付款额

从 `Annual Interest Rate:JTextField` 中获取年利率

计算贷款期限分别为 2 年、3 年、4 年和 5 年的月度支付额

显示结果

catch `NumberFormatException` 异常

显示错误消息对话框

当用户调用 `calculateMonthlyPayment` 方法时

根据贷款额、月利率和贷款期限(以月为单位)计算月度支付额

既然读者已探试了改进的购车还贷计算器应用程序并研究了它的伪代码表示,下面,我们使用一张 ACE 表,帮助读者把这个伪代码转换成 Java 实现。图 24.8 列出了该应用程序中相应的操作、组件以及事件,帮助读者最终完成属于自己的应用程序。


操作	组件 / 类	事件
 标记应用程序中的组件	priceJLabel	当应用程序运行时
	downPaymentJLabel	
清除原先 JTextArea 中的文本	interestRateJLabel	
	calculateJButton	当用户点击 Calculate JButton 时
try	monthlyPaymentsJTextArea	
从 Price:JTextField 中获取汽车价格	priceJTextField	
从 Down Payment:JTextField 中获取首期付款额	downPaymentJTextField	
从 Annual Interest Rate:JTextField 中获取年利率	interestRateJTextField	
计算贷款期限分别为 2 年、3 年、4 年和 5 年的		
月度支付额显示结果	monthlyPaymentsJTextArea	
catch NumberFormatException 类	JOptionPane	
显示错误消息对话框		调用calculateMonthlyPayment方法
根据贷款额、月利率和贷款期限（以月为单位）		
计算月度支付额		

图 24.8 改进的购车还贷计算器应用程序的 ACE 表

到目前为止，读者分析了改进的购车还贷计算器应用程序中的组件，下面，将学习如何在应用程序代码中放置异常处理的代码。

处理 NumberFormatException 异常

- 1. 将模板复制到工作目录中 将 C:\Examples\Tutorial24\TemplateApplication\EnhancedCarPayment 目录复制到 C:\SimplyJava 目录中。
- 2. 打开模板应用程序 在自己的文本编辑器中打开 CarPayment.java 文件。
- 3. 研究代码 在图 24.9 中，用于读取 JTextField 中整数的语句（参见第 120 行至第 124 行）通过 Integer.parseInt 方法将一个 String 转换成一个 int 值，从 JTextField 中读取 double 值的语句（参见第 127 行至第 128 行）则使用 Double.parseDouble 方法将 String 转换成一个 double 值。

若 parseInt 和 parseDouble 方法中的参数不是一个合法的整数或 double 数时，会抛出 NumberFormatException 异常。作为 RuntimeException 子类的 NumberFormatException 类，其异常是在应用程序无法将一个 String 转换为所需要的数值类型时（如 int 或 double）抛出的。其中，当应用程序试图把一个未表示成整数的字符串（如 "6000.05" 或 "600p"）转换为一个 int 值时，方法 parseInt 会抛出 NumberFormatException 异常。同样，当应用程序试图把一个未表示成浮点数的字符串（如 7.5%）转换为一个 double 值时，方法 parseDouble 会抛出 NumberFormatException 异常。然而，如果传递给 parseDouble 方法的参数是一个整数形式的字符串（如 "7"），则不会抛出 NumberFormatException 异常，因为整数也属于一种有效的 double 值。一旦数据被读入并且 JTextField 中的内容已转换为 int 值或 double 值时，可以通过调用 processData 方法确定出月度支付额（参见第 131 行）。

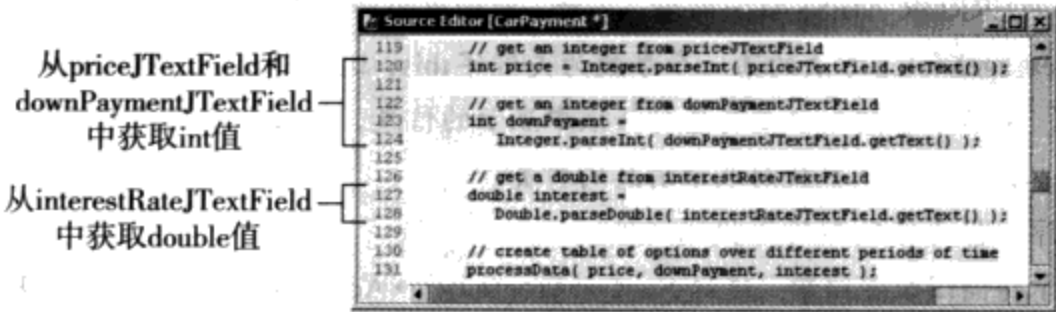


图 24.9 获取用户输入

4. 添加 try 语句 将图 24.10 中第 122 行至第 123 行和第 138 行添加到程序代码中。该 try 语句试图从 Price: 和 Down payment:JTextField 中获取一个 int 值, 从 Annual interest rate:JTextField 中获取一个 double 值。在添加完图 24.10 中高亮显示的代码以后, 需要将第 125 行至第 137 行的代码进行缩进处理。第 125 行至第 134 行的代码需要放置在一个 try 语句块中, 因为它们可能会有异常抛出。第 137 行即使不抛出异常也应包含在 try 语句块中, 因为我们不希望在产生异常的时候去执行它。如果第 137 行以前的任意一行发生异常, 那么将跳过该行代码。

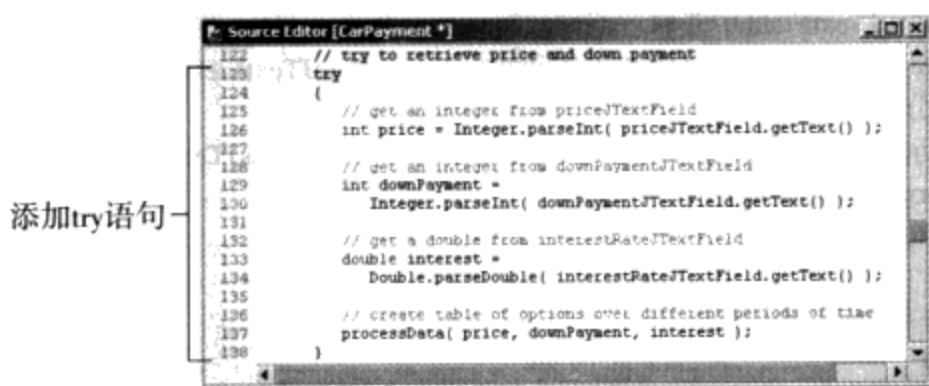


图 24.10 在 calculateJButton 事件处理程序中添加 try 语句

5. 添加 catch 语句块 本例中 try 语句块的后面只需要一个 catch 语句块, 因为只有一种异常类型 (NumberFormatException) 会被抛出。将图 24.11 中第 139 行至第 142 行添加到程序代码中。第 139 行指明了该语句块所能捕获的异常种类。这样, 通过异常处理程序可以防止用户在需要输入整数的 JTextField 中输入小数, 或者是在需要输入数字值的 JTextField 中输入非数字数据 (例如, 将 6000 输入为 600p)。如果用户未能在 priceJTextField 中输入一个 int 值, 第 126 行会抛出一个 NumberFormatException 异常。同样, 如果用户未能在 downPaymentJTextField 中输入一个 int 值, 第 130 行也会抛出一个 NumberFormatException 异常, 而如果用户未在 interestJTextField 中输入一个 double 值 (或 int 值), 则第 134 行 (如图 24.10 所示) 将抛出一个 NumberFormatException 异常。将处理这类异常的声明放在与该 try 语句块相对应的 catch 语句块中。如果读者试图捕获多个错误, 则可在 try 语句块的后面添加多个 catch 语句块。

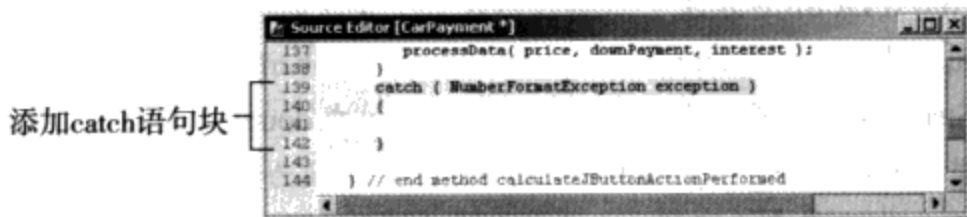


图 24.11 添加 catch 语句块

6. 显示一个错误消息对话框 将图 24.12 中第 138 行至第 142 行添加到应用程序中。如果产生 NumberFormatException 异常, 则通知用户并重新输入正确的数据。为实现这一功能, 读者将利用 JOptionPane 显示一个消息对话框 (如图 24.3 所示)。由于处理的是与异常有关的操作, 可以使用 JOptionPane 的 ERROR_MESSAGE 常量。另外, 前面曾经讲过 this 关键字是指当前对象, 在本例中, 传递给 showMessageDialog 方法的第一个参数为关键字 this, 这样错误消息对话框将出现在应用程序窗口的中央位置。第二个参数表示在消息对话框中显示的信息, 第三个参数用来设置消息对话框标题栏内所显示的文本, 最后一个参数则表示消息的种类。
7. 保存应用程序 保存修改后的源代码文件。
8. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。通过键入 cd C:\SimplyJava\EnhancedCarPayment 进入到当前工作目录中。
9. 编译应用程序 键入 javac CarPayment.java 编译该应用程序。

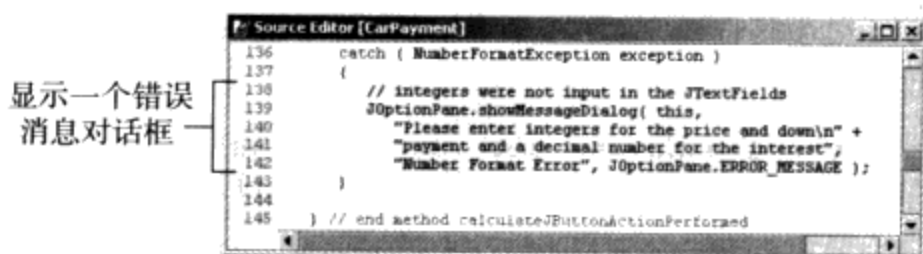


图 24.12 显示一个错误消息对话框

10. 运行应用程序 若能正确编译应用程序，通过键入 `java CarPayment` 来运行它。图 24.13 显示了用户输入无效数据时的运行结果。对应用程序重新进行测试，观察结果是否正确。点击 OK JButton 关闭此消息对话框。

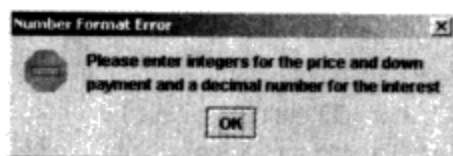


图 24.13 产生 NumberFormatException 时所显示的错误消息对话框

11. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
12. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

图 24.14 中给出了改进的购车还贷计算器应用程序的源代码。本教程中，凡需要添加、查看或者修改的代码，均进行了突出显示。

```

1 // Tutorial 24: CarPayment.java
2 // This application uses exception-handling to handle invalid input.
3 import java.awt.*;
4 import java.text.DecimalFormat;
5 import java.awt.event.*;
6 import javax.swing.*;
7
8 public class CarPayment extends JFrame
9 {
10     // JLabel and JTextField for price
11     private JLabel priceJLabel;
12     private JTextField priceJTextField;
13
14     // JLabel and JTextField for down payment
15     private JLabel downPaymentJLabel;
16     private JTextField downPaymentJTextField;
17
18     // JLabel and JTextField for interest rate
19     private JLabel interestRateJLabel;
20     private JTextField interestRateJTextField;
21
22     // JButton to calculate the monthly payments
23     private JButton calculateJButton;
24
25     // JTextArea to display the monthly payments
26     private JTextArea monthlyPaymentsJTextArea;
27
28     // no-argument constructor
29     public CarPayment()
30     {
31         createUserInterface();
32     }
33

```

```
34 // create and position GUI components; register event handlers
35 private void createUserInterface()
36 {
37     // get content pane and set layout to null
38     Container contentPane = getContentPane();
39
40     // enable explicit positioning of GUI components
41     contentPane.setLayout( null );
42
43     // set up priceJLabel
44     priceJLabel = new JLabel();
45     priceJLabel.setBounds( 40 , 24 , 80 , 21 );
46     priceJLabel.setText( "Price:" );
47     contentPane.add( priceJLabel );
48
49     // set up priceJTextField
50     priceJTextField = new JTextField();
51     priceJTextField.setBounds( 184, 24 , 56 , 21 );
52     priceJTextField.setHorizontalAlignment( JTextField.RIGHT );
53     contentPane.add( priceJTextField );
54
55     // set up downPaymentJLabel
56     downPaymentJLabel = new JLabel();
57     downPaymentJLabel.setBounds( 40 , 56 , 96 , 21 );
58     downPaymentJLabel.setText( "Down payment:" );
59     contentPane.add( downPaymentJLabel );
60
61     // set up downPaymentJTextField
62     downPaymentJTextField = new JTextField();
63     downPaymentJTextField.setBounds( 184, 56 , 56 , 21 );
64     downPaymentJTextField.setHorizontalAlignment(
65         JTextField.RIGHT );
66     contentPane.add( downPaymentJTextField );
67
68     // set up interestRateJLabel
69     interestRateJLabel = new JLabel();
70     interestRateJLabel.setBounds( 40 , 88, 120, 21 );
71     interestRateJLabel.setText( "Annual interest rate:" );
72     contentPane.add( interestRateJLabel );
73
74     // set up interestRateJTextField
75     interestRateJTextField = new JTextField();
76     interestRateJTextField.setBounds( 184, 88, 56 , 21 );
77     interestRateJTextField.setHorizontalAlignment(
78         JTextField.RIGHT );
79     contentPane.add( interestRateJTextField );
80
81     // set up calculateJButton
82     calculateJButton = new JButton();
83     calculateJButton.setBounds( 92 , 128 , 94, 24 );
84     calculateJButton.setText( "Calculate" );
85     contentPane.add( calculateJButton );
86     calculateJButton.addActionListener(
87
88         new ActionListener() // anonymous inner class
89         {
90             // event handler called when calculateJButton is clicked
91             public void actionPerformed((ActionEvent event) )
92             {
93                 calculateJButtonActionPerformed( event );
94             }
95         }
```

```

95
96     } // end anonymous inner class
97
98     ); // end call to addActionListener
99
100    // set up monthlyPaymentsJTextArea
101    monthlyPaymentsJTextArea = new JTextArea();
102    monthlyPaymentsJTextArea.setEditable( false );
103    monthlyPaymentsJTextArea.setBounds( 28 , 168 , 232, 90 );
104    contentPane.add( monthlyPaymentsJTextArea );
105
106    // set properties of application's window
107    setTitle( "Car Payment Calculator" ); // set title-bar string
108    setSize( 288, 302 ); // set window size
109    setVisible( true ); // display window
110
111 } // end method createUserInterface
112
113 // calculate the monthly car payments
114 private void calculateJButtonActionPerformed((ActionEvent event) )
115 {
116     // clear JTextArea
117     monthlyPaymentsJTextArea.setText( "" );
118
119     // try to retrieve price and down payment
120     try // 将可能产生异常的代码封装在一个try语句块中
121     {
122         // get an integer from priceJTextField
123         int price = Integer.parseInt( priceJTextField.getText() );
124
125         // get an integer from downPaymentJTextField
126         int downPayment =
127             Integer.parseInt( downPaymentJTextField.getText() );
128
129         // get a double from interestRateJTextField
130         double interest =
131             Double.parseDouble( interestRateJTextField.getText() );
132
133         // create table of options over different periods of time
134         processData( price, downPayment, interest );
135     } // try语句块的结束位置
136     catch ( NumberFormatException exception ) // 添加一个处理
137     { // NumberFormatException
138         // integers were not input in the JTextFields // 异常的catch语句块
139         JOptionPane.showMessageDialog( this ,
140             "Please enter integers for the price and down\n" +
141             "payment and a decimal number for the interest" ,
142             "Number Format Error" , JOptionPane.ERROR_MESSAGE );
143     }
144
145 } // end method calculateJButtonActionPerformed
146
147 // process entered data and calculate payments over
148 // each time interval
149 private void processData( int price, int downPayment,
150     double interest )
151 {
152     // calculate loan amount and monthly interest
153     int loanAmount = price - downPayment;
154     double monthlyInterest = interest / 1200;

```

```
155
156     // format to display monthlyPayment in currency format
157     DecimalFormat dollars = new DecimalFormat( "$0.00" );
158
159     int years = 2 ; // repetition counter
160
161     // add header JTextArea
162     monthlyPaymentsJTextArea.append( "Months\tMonthly Payments" );
163
164     // while years is less than or equal to five years
165     while ( years <= 5 )
166     {
167         // calculate payment period
168         int months = 12 * years;
169
170         // get monthlyPayment
171         double monthlyPayment = calculateMonthlyPayment(
172             monthlyInterest, months, loanAmount );
173
174         // insert result into JTextArea
175         monthlyPaymentsJTextArea.append( "\n" + months + "\t" +
176             dollars.format( monthlyPayment ) );
177
178         years++; // increment counter
179     } // end while
180 } // end method processData
181
182 // calculate monthlyPayment
183 private double calculateMonthlyPayment( double monthlyInterest,
184     int months, int loanAmount )
185 {
186     double base = Math.pow( 1 + monthlyInterest, months );
187     return loanAmount * monthlyInterest / ( 1 - ( 1 / base ) );
188 } // end method calculateMonthlyPayment
189
190 // main method
191 public static void main( String[] args )
192 {
193     CarPayment application = new CarPayment();
194     application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
195 } // end method main
196
197 } // end class CarPayment
```

图 24.14 改进的购车还贷计算器应用程序的源代码

自测题

1. 如果试图捕获多个错误, 可在 _____ 语句块的后面使用多个 _____ 语句块。
a) try catch b) catch try c) finally try d) 以上答案都不对
2. 将需要处理的异常类作为 _____ 语句块的一个参数来进行声明。
a) try b) catch c) finally d) 以上都不对

答案: 1) b 2) b

24.6 小结

通过对本教程的学习,读者掌握了有关异常处理的概念以及如何通过 Java 处理异常。利用 try 语句块和 catch 语句块可以处理应用程序中的异常,而通过使用 throws 语句可以在方法中抛出异常。另外,读者还了解了如何使用 throw 关键字重抛一个未在 catch 语句块中处理的异常。接着,学习了异常类的继承结构。运用 Java 中的异常处理技术,改进了购车还贷计算器应用程序,使该程序在计算汽车价格的同时执行输入检查功能。即利用 try 语句块将可能抛出 NumberFormatException 异常的语句进行封装并使用 catch 语句块实现对 NumberFormatException 异常的处理。

在下一个教程中,将学习如何表示计算机中的数据,还将了解到有关文件和流的概念,以及如何通过按顺序存取的文件存储数据,并且还将使用 finally 语句块关闭用于读取文件或写入文件的流。

技术小结

处理异常

- 将可能产生异常及产生异常时不需要执行的代码封装在一个 try 语句块中。
- try 语句块的后面可跟多个 catch 语句块。每个 catch 语句块将作为一个特定异常类型的异常处理程序。

关键术语

catch 语句块 也称异常处理程序。当应用程序中相应的 try 语句块检测到一个异常并抛出此 catch 语句块所声明的异常类型时,执行该 catch 语句块。

捕获或声明要求 产生检查型异常的代码需封装在一个 try 语句块中并为其提供相应的一个 catch 语句块,或者产生检查型异常代码的方法在未能提供相应的事件处理程序时,必须提供一个用于声明该检查型异常的 throws 语句。

检查型异常 一种非 RuntimeException 子类的异常类型,该异常必须满足捕获或声明要求。

Error 类 代表 Java 运行时系统中所发生的异常,该异常通常不需要在应用程序中捕获。

异常 应用程序执行过程中所发生的一个错误。

Exception 类 表示在 Java 应用程序中出现的可被捕获并处理的异常。

异常处理程序 当应用程序检测并抛出一个异常时,所执行的一个语句块(位于 catch 语句块内)。

异常处理机制 创建解决(或处理)应用程序执行过程中所产生问题的一个过程。

异常的参数 指定异常处理程序中所能处理的异常类型。

finally 语句块 确保 try 语句块在执行时一定能够执行的一个语句块。通常情况下,finally 语句块用来对资源进行释放。

NumberFormatException 类 RuntimeException 类的一个子类,该异常是在应用程序不能将一个字符串转换为所需要的数值类型时(如 int 或 double)抛出的。

异常的重抛 利用 throw 关键字将 catch 语句块中的异常处理(或其中的一部分)推迟到另一个 catch 语句块中。

RuntimeException 类 应用程序执行过程中所抛出的非检查型异常的超类。

Throwable 类 所有异常和错误的超类。

throw 关键字 catch 语句块中用来重抛异常的一个关键字。

抛出异常 方法执行过程中如果产生一个该方法未进行处理的问题,则抛出一个相应的异常。

throws 语句 位于方法参数列表之后、方法体之前,指明该方法抛出但未进行处理的异常。该语句中包含了一个由逗号分隔的异常列表,表示方法执行时若有问题产生,则将抛出异常列表中所给出的异常。

try 语句块 该语句块中包含可能产生异常的语句以及产生异常时不应执行的语句。

try 语句 由关键字 try,一对大括号({}),一个或多个 catch 语句块,一个可以选用的 finally 语句块所组成的语句。

非捕获型异常 一种可以不进行异常处理的异常。非捕获型异常会终止应用程序的执行。

非检查型异常 代表 `RuntimeException` 子类的一种异常类型。这种异常是在应用程序执行时抛出的。

Java 类库索引

Error 此类代表 Java 运行时系统发生的异常，通常情况下，应用程序不需要捕获它们。

Exception 此类代表发生在 Java 应用程序中的异常，这种异常需要在应用程序中捕获。

NumberFormatException `RuntimeException` 类的一个子类，当应用程序不能将一个字符串转换成所需要的数值类型时（如 `int` 或 `double`）抛出。

RunTimeException 所有非检查型异常的超类，在应用程序执行时抛出。

Throwable 所有异常和错误类的共有超类。

习题

选择题

- 24.1 程序执行过程中对异常的处理称为异常的_____。
a) 检测 b) 处理 c) 解决 d) 调试
- 24.2 在_____之后总有至少一个 `catch` 语句块或一个可以选用的 `finally` 语句块。
a) `if` 语句 b) `throws` 语句块 c) `try` 语句块 d) 以上答案都不对
- 24.3 调用方法 `Integer.parseInt("123.45")`；将抛出_____。
a) `NumberFormatException` b) `ParsingException`
c) `ArithmeticException` d) 以上答案都不对
- 24.4 如果 `try` 语句块中没有抛出异常，则_____。
a) 跳过所有的 `catch` 语句块 b) 执行所有的 `catch` 语句块
c) 产生一个错误 d) 抛出默认的异常
- 24.5 每个 `catch` 块都需指定一个_____，表示异常处理程序所能处理的异常类型。
a) `try` 语句块 b) 参数 c) 错误处理程序 d) 抛出程序
- 24.6 `try` 语句块可以_____。
a) 拥有惟一的一个 `catch` 语句块 b) 多个 `finally` 语句块
c) 一个或多个 `catch` 语句块 d) 以上答案都不对
- 24.7 所有 `RuntimeException` 子类的异常都属于_____。
a) 致命错误 b) 逻辑错误 c) 检查型异常 d) 非检查型异常
- 24.8 _____是所有 `Exception` 和 `Error` 类的共同超类。
a) `Throwable` b) `CheckedException` c) `Catchable` d) `RuntimeException`
- 24.9 _____类及其子类所表示的异常通常不需要捕获。
a) `NumberFormatException` b) `Exception` c) `RuntimeException` d) `Error`
- 24.10 如果在 `try` 语句块中抛出异常或者未抛出异常，都需要执行_____。
a) `catch` 语句块 b) `finally` 语句块 c) 异常处理程序 d) 以上答案都正确

练习题

- 24.11 （改进的每加仑英里数应用程序）通过修改每加仑英里数应用程序（练习 12.13 所示），添加一个用于处理当 `JTextField` 中的字符串被转换为 `double` 时，产生的 `NumberFormatException` 异常（如图 24.15 所示）。为计算每加仑汽油所行驶的英里数，原应用程序允许用户输入汽车已行驶的英里数及油箱内的汽油量。
- a) 将模板复制到工作目录中 将 `C:\Examples\Tutorial24\Exercises\EnhancedMilesPerGallon` 目录复制到 `C:\SimplyJava` 目录中。
- b) 打开模板文件 在自己的文本编辑器中打开 `MilesPerGallon.java` 文件。

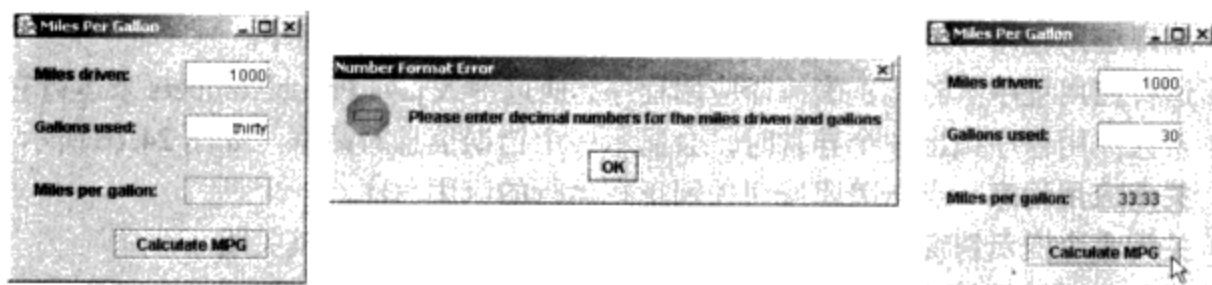


图 24.15 改进的每加仑英里数应用程序的 GUI

- c) 添加 try 语句块 找到 calculateMPGJButtonActionPerformed 方法(该方法位于 createUserInterface 方法之后), 将第 113 行至第 120 行封装到一个 try 语句块中。
 - d) 添加 catch 语句块 在步骤(c)中所添加 try 语句块的后面, 添加一个用于捕获 NumberFormatException 异常的 catch 语句块。在 catch 语句块的内部, 添加相应的代码, 将一个错误消息对话框居中并显示在应用程序运行窗口的上面。
 - e) 保存应用程序 保存修改后的源代码文件。
 - f) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\EnhancedMilesPerGallon` 进入到当前工作目录中。
 - g) 编译应用程序 键入 `javac MilesPerGallon.java` 编译该应用程序。
 - h) 运行程序应用 若能正确编译应用程序, 键入 `java MilesPerGallon` 来运行它。当完成后的每加仑英里数应用程序抛出一个异常时, 会显示一个错误消息对话框(如图 24.15 所示)。
 - i) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
 - j) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。
- 24.12 (改进的素数应用程序)通过修改素数应用程序(练习 12.13 所示)添加一个用于处理因 JTextField 中的字符串被转换为 int 值时所产生的 NumberFormatException 异常(如图 24.16 所示)。原应用程序通过接收两个数(代表上下界)确定出一个指定范围内(含上下界)的所有素数。所谓素数是指大于 1 且只能被 1 或其本身整除的 int 型整数。例如, 2, 3, 5, 7 是素数, 4, 6, 8 和 9 则不是素数。



图 24.16 改进的素数应用程序的 GUI

- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial24\Exercises\EnhancedPrimeNumbers 目录复制到 C:\SimplyJava 目录中。
- b) 开模板文件 在自己的文本编辑器打开 PrimeNumbers.java 文件。
- c) 添加 try 语句块 找到 calculatePrimesJButtonActionPerformed 方法, 该方法位于 createUserInterface 方法之后。将第 116 行至第 123 行封装在一个 try 语句块中。
- d) 添加 catch 语句块 在步骤(c)中所添加 try 语句块的后面, 添加一个用于捕获 NumberFormatException 异常的 catch 语句块。在 catch 语句块的内部, 添加相应的代码, 将一个错误消息对话框居中显示在应用程序运行窗口的上面。
- e) 保存程序 保存修改后的源代码文件。
- f) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\EnhancedPrimeNumbers` 进入到当前工作目录中。

- g) **编译应用程序** 键入 `javac PrimeNumbers.java` 编译该应用程序。
 - h) **运行应用程序** 若能正确编译应用程序, 通过键入 `java PrimeNumbers` 来运行它。当完成后的素数应用程序抛出一个异常时, 会显示一个错误信息对话框 (如图 24.16 所示)。
 - i) **关闭应用程序** 点击关闭按钮关闭正在运行的应用程序。
 - j) **关闭命令提示符窗口** 点击命令提示符窗口的关闭按钮将其关闭。
- 24.13 (改进的简单计算器应用程序) 通过修改简单计算器应用程序 (如练习 5.13 所示) 添加一个用于处理因 `TextField` 中的字符串被转换为 `int` 时所产生的 `NumberFormatException` 异常, 以及当执行除法运算时所产生的 `ArithmeticException` 异常 (如图 24.17 所示)。此应用程序仍然能够用于执行简单的加、减、乘、除运算。
- a) **将模板复制到工作目录中** 将 `C:\Examples\Tutorial24\Exercises\EnhancedSimpleCalculator` 目录复制到 `C:\SimplyJava` 目录中。
 - b) **打开模板文件** 在自己的文本编辑器中打开 `SimpleCalculator.java` 文件。
 - c) **给 `addJButtonActionPerformed` 方法添加一个 `try` 语句块** 找到 `addJButtonActionPerformed` 方法 (该方法位于 `secondNumberJTextFieldKeyPressed` 方法的后面), 将此方法的方法体 (第 220 行至第 229 行) 封装在一个 `try` 语句块中。
 - d) **给 `addJButtonActionPerformed` 方法添加一个 `catch` 语句块** 在 `addJButtonActionPerformed` 方法内 `try` 语句块的后面, 添加一个用于捕获 `NumberFormatException` 异常的 `catch` 语句块。在 `catch` 语句块的内部, 添加相应的代码, 将一个错误消息对话框居中显示在应用程序运行窗口的上面。

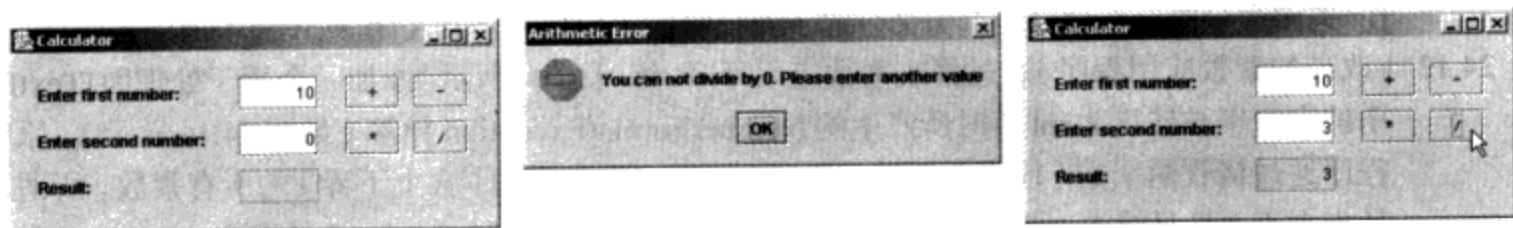


图 24.17 改进的简单计算器应用程序

- e) **给 `subtractJButtonActionPerformed` 方法添加一个 `try` 语句块** 找到 `subtractJButtonActionPerformed` 方法 (位于 `addJButtonActionPerformed` 方法的后面), 将此方法的方法体封装在一个 `try` 语句块中。
- f) **给 `subtractJButtonActionPerformed` 方法中添加一个 `catch` 语句块** 紧接 `try` 语句块, 添加一个用于捕获 `NumberFormatException` 异常的 `catch` 语句块。在 `catch` 语句块的内部, 添加相应的代码, 将一个错误消息对话框居中显示在应用程序运行窗口的上面。
- g) **给 `multiplyJButtonActionPerformed` 方法添加一个 `try` 语句块** 找到 `multiplyJButtonActionPerformed` 方法 (位于 `subtractJButtonActionPerformed` 方法的后面), 将此方法的方法体封装在一个 `try` 语句块中。
- h) **给 `multiplyJButtonActionPerformed` 方法添加一个 `catch` 语句块** 紧接 `try` 语句块, 添加一个用于捕获 `NumberFormatException` 异常的 `catch` 语句块。在 `catch` 语句块的内部, 添加相应的代码, 将一个错误消息对话框居中显示在应用程序运行窗口的上面。
- i) **给 `divideJButtonActionPerformed` 方法添加一个 `try` 语句块** 找到 `divideJButtonActionPerformed` 方法 (位于 `multiplyJButtonActionPerformed` 方法的后面), 将此方法的方法体封装在一个 `try` 语句块中。
- j) **给 `divideJButtonActionPerformed` 方法添加一个 `catch` 语句块** 紧接 `try` 语句块, 添加一个用于捕获 `NumberFormatException` 异常的 `catch` 语句块。在 `catch` 语句块的内部, 添加相应的代码, 将一个错误消息对话框居中显示在应用程序运行窗口的上面。
- k) **给 `divideJButtonActionPerformed` 方法再添加一个 `catch` 语句块** 紧接第一个 `catch` 语句块, 再添加一个用于捕获 `ArithmeticException` 异常的 `catch` 语句块。整数运算中若除数为 0, 则会

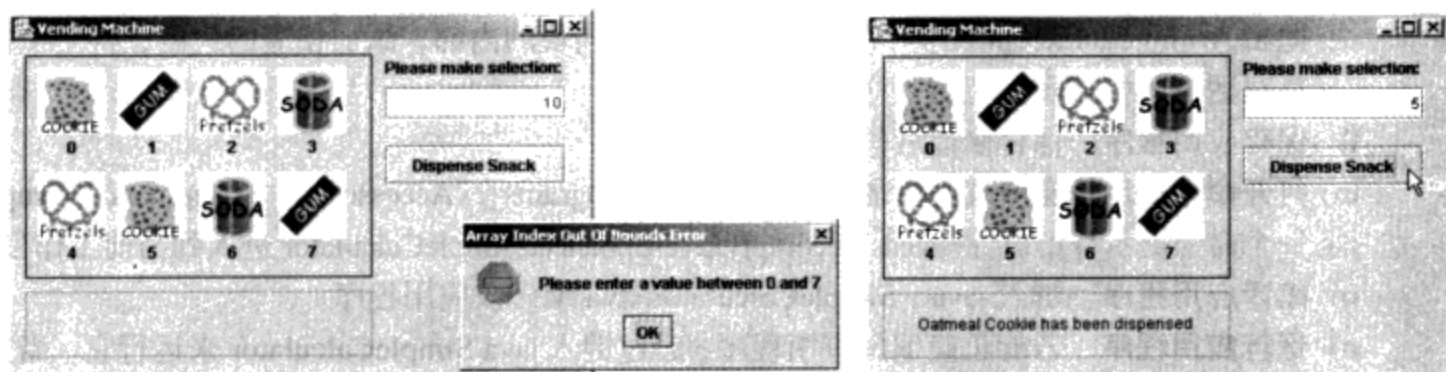


图 24.18 改进的自动售货机应用程序的 GUI

- b) 打开模板文件 在自己的文本编辑器中打开 VendingMachine.java 文件。
- c) 添加一个 try 语句块 找到 dispenseJButtonActionPerformed 方法（位于 createUserInterface 方法的后面），将第 236 行至第 240 行封装在一个 try 语句块中。
- d) 添加一个 catch 语句块 接着步骤(c)中所添加的 try 语句块，再添加一个用于捕获 NumberFormatException 异常的 catch 语句块。在 catch 语句块的内部，添加相应的代码，将一个错误消息对话框居中显示在应用程序运行窗口的上面。
- e) 添加第二个 catch 语句块 接着步骤(d)中所添加的 catch 语句块，再添加一个用于捕获 ArrayIndexOutOfBoundsException 异常的 catch 语句块。ArrayIndexOutOfBoundsException 异常是当应用程序试图通过一个无效索引来访问数组时出现的。在该 catch 语句块的内部，添加相应的代码，将一个错误消息对话框居中显示在应用程序运行窗口的上面。
- f) 保存应用程序 保存修改后的源代码文件。
- g) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口，键入 `cd C:\SimplyJava\VendingMachine` 进入到当前工作目录中。
- h) 编译应用程序 键入 `javac VendingMachine.java` 编译该应用程序。
- i) 运行应用程序 若能正确编译应用程序，通过键入 `java VendingMachine` 来运行它。当完成后的自动售货机应用程序抛出一个 NumberFormatException 异常时，会显示一个类似前面练习中所出现的 JOptionPane。当用户试图访问超出 0~7 范围的一个元素时，则出现一个如图 24.18 所示的 JOptionPane。
- j) 关闭应用程序 点击关闭按钮关闭正在运行的程序。
- k) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。



教程 25 票务信息查询应用程序

介绍按顺序存取的文件

教学目标

在本教程中，读者将学到以下内容：

- 文件的创建、读取、写入及更新
- 计算机数据分级的理解
- 熟悉按顺序存取文件的处理过程
- 使用具有 `BufferedReader` 功能的 `FileReader` 类从按顺序存取的文件中读取一行文本
- 使用具有 `PrintWriter` 功能的 `FileWriter` 类向按顺序存取的文件中写入文本
- 在异常处理机制中使用 `finally` 语句块确保特定操作的执行

读者曾使用变量和数组临时存储数据——一旦方法和应用程序终止运行，这些数据将会丢失。如果想长时间地保存数据，可以考虑使用文件。文件是一个被赋予名字（如 `data.txt` 或 `Welcome.java`）的数据集合。文件中的数据会在其创建的应用程序终止之后仍然存在。通常称这一类型的数据为持久数据。计算机中的文件存储在辅助存储介质上，辅助存储介质包括磁盘（如计算机硬盘）、光盘（如 `CD-ROM` 或 `DVD`）、磁带（类似于盒式卡带），等等。

文件处理包括文件的创建、读取及写入，它是 Java 的一项重要功能。利用文件处理，Java 可开发出用于含大量持久数据的商用应用程序。在本教程中，读者将学习有关按顺序存取文件的知识，该文件中所包括的信息能够按其写入的顺序重新读取出来。读者将创建一个活动录入应用程序，在该应用程序中学习如何创建、打开以及写入一个按顺序存取的文件。活动录入应用程序允许用户创建或打开一个文本文件（由人们所能识别的字符组成），用户可以输入日期、时间以及与社团活动（如音乐会或体育比赛等）相关的描述。

随后，还将学习如何构建票务信息查询应用程序从文件中读取数据。该应用程序可以显示由活动录入应用程序所创建的文件中的数据。在这个过程中，读者会再次使用到二维数组，利用二维数组存储文件中读出的数据。

25.1 探试活动录入应用程序和票务信息查询应用程序

许多社团及企业利用计算机让员工和顾客了解即将举行的一些活动的信息，如电影、音乐会、体育活动等。本教程中创建的活动录入应用程序可以把社团内举办的活动信息写入一个按顺序存取的文件中。该应用程序需满足以下需求：

应用程序需求分析

为创建一个能够从文件中读取信息的应用程序，需要首先创建一个用于向文件中写入活动信息的应用程序。该应用程序为用户提供了有关日期、时间、名称及相关活动信息描述的输入组件。同时，还能够给出此文件的位置和名称。

读者将创建的票务信息查询应用程序用于显示存储在活动录入应用程序所创建的文件中的数据。票务信息查询应用程序需满足以下需求：

应用程序需求分析

某城镇希望通过开发一个应用程序，让全体居民查看当月社团所举办的活动，这些活动如音乐会、体育赛事、电影及其他娱乐，等等。所有活动已通过应用程序录入到了 calendar.txt 文件中。当用户选定一个日期时，应用程序需给出当天是否有活动安排。同时，还应列出所有已安排的活动并允许用户选择一项特定的活动。之后，应用程序应显示出时间、价格及关于该项活动的一个简要描述。如果用户选择的当天没有活动安排，则应向用户发出相应的通知信息。

票务信息查询应用程序允许用户从 JSpinner GUI 组件中选择一个日期。为方便起见，日期为一个 1 到 31（含边界值）之间的整数。假定所有活动只限于同一年内的一个月，因此，用户需要关心的只是本月中具体的那一天。同时，假定该月有 31 天。应用程序能够打开一个文本并读出其中的内容，以此显示出与该所选日期活动安排相关的信息。读者将以这个完成后的应用程序的探试作为起点。之后，学习一些 Java 技术并最终创建一个属于自己的应用程序。



探试票务信息查询和活动录入应用程序

1. 定位到完成后的应用程序 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\Examples\Tutorial25\CompletedApplication\TicketInformation` 将目录改变到这个完成后的票务信息查询应用程序的目录下面。
2. 运行票务信息查询应用程序 在命令提示符窗口下通过键入 `java TicketInformation` 运行该应用程序（如图 25.1 所示）。在应用程序 GUI 的上方，有一个供用户选择日期的 JSpinner 组件。JSpinner 组件当前显示的值为 1，表明该月第一天。另外，还有一个列有 JSpinner 内某一天活动的 JComboBox 组件。若 JComboBox 中显示 No Events，则表示当天并没有活动安排。
3. 获取活动信息 选择该月第 19 天。注意观察 JComboBox 中现在显示出一个活动信息（如图 25.2 所示）。与此活动相关的时间、价格及其描述出现在了 Description:JTextArea 中。点击 Pick an event: JComboBox 中的向下箭头，可以看到，该天只有一项活动：Film Festival。如果一天内有多项活动，那么，这些活动都将列入 JComboBox 中。Description:JTextArea 中只显示列表内与第一项活动相关的信息。要查看其他活动，只需从 Pick an event:JComboBox 中选择相应的活动名称。下面将举例说明。



图 25.1 票务信息查询应用程序的 GUI



图 25.2 显示活动信息的票务信息查询应用程序

4. 查看其他活动 另选择其他日期（如第 4 天、第 5 天或第 28 天）查看结果。第 4 天显示没有活动，第 5 天显示有一个业余歌手演唱会，第 28 天显示了一条关于城市旅游的信息。
5. 关闭正在运行的应用程序 点击关闭按钮关闭正在运行的应用程序，保留命令提示符窗口。
6. 通过活动录入应用程序添加活动 在命令提示符窗口中，键入 `cd C:\Examples\Tutorial25\CompletedApplication\WriteEvent` 将目录改变到完成后的活动录入应用程序的目录下面。在命令提示符

窗口中键入 java WriteEvent 运行该应用程序(如图 25.3 所示)。将出现一些供用户输入日期、时间、价格、名称及活动描述的输入组件。Time:JSpinner 最初显示为当前时间(读者将在本教程中学习如何读取系统时间),还有三个 JButton,但只有 Open File... JButton 处于启用状态。注意,在写入有关活动的

7. 打开一个文件 票务信息查询应用程序所使用的文件是 calendar.txt,该文件位于 TicketInformation 目录下。为了向文件中增加活动,需要先把它打开。点击 Open File... JButton 打开 Open File for Write Event 对话框(如图 25.4 所示)。浏览到 C:\Examples\Tutorial25\CompletedApplication\TicketInformation 目录,选择 calendar.txt 文件。然后,点击 Open JButton。

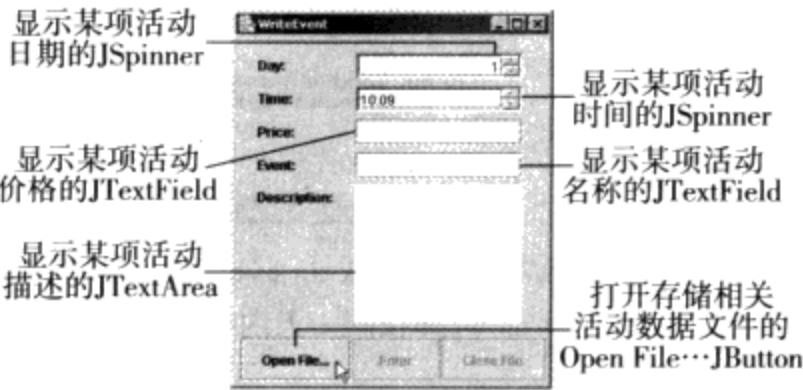


图 25.3 通过活动录入应用程序存储相关活动数据

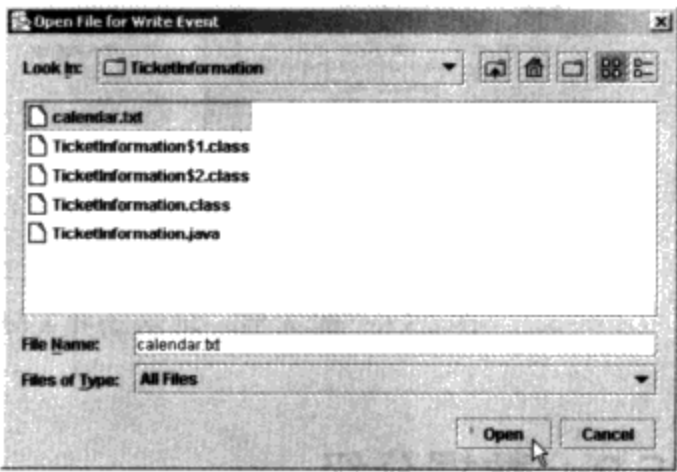


图 25.4 提供选择存储活动项数据文件的对话框

8. 输入一项活动 返回到应用程序主窗口中。注意, Open File... JButton 已被禁用(因为文件已经打开), Enter 和 Close File JButton 则被启用。按照图 25.5 左侧的显示输入其中一项活动的信息,然后点击 Enter JButton。在向 Description:JTextArea 中录入信息时不要按回车键,以免向 calendar.txt 文件中添加额外的文本行。所有录入的数据都将写入上一步骤中所指定的 calendar.txt 文件内。之后,录入该活动价格、名称和描述的 JTextField 组件中的文本被清除掉了(如图 25.5 的右侧显示)。留下的一个显示时间的 JSpinner 可让用户通过点击上/下箭头查询时间(24 小时制)。同样,很容易在 JSpinner 中显示出分钟。

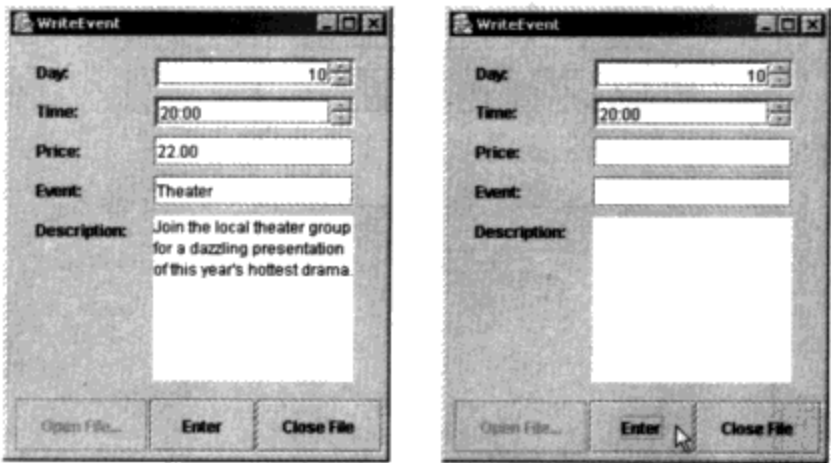


图 25.5 为 calendar.txt 中添加活动

9. 关闭文件 按照图 25.6 左侧的显示输入另一项活动,然后,点击 Enter JButton。随后点击 Close File JButton(如图 25.6 右侧的显示),文件 calendar.txt 将被关闭。可以看到,用于录入价格、名称及描述的 JTextField 中的文本将再次被清除,而 Open File... JButton 则被启用,以便重新打开一个文件或再次打开 calendar.txt。目前,由于未打开一个文件,所以 Enter 和 Close File JButton 都被禁用。
10. 关闭正在运行的应用程序 点击窗口上的关闭按钮,关闭正在运行的应用程序。
11. 查看新的事件 键入 cd C:\Examples\Tutorial25\CompletedApplication\TicketInformation 将目录改变到这个完成后的票务信息查询应用程序的目录下面。在命令提示符窗口下,键入 java TicketInformation 运行该应用程序。目前,读者已添加了第 10 天和第 19 天的活动信息。依次选择这两天,查看所对应

活动项目的显示情况。图 25.2 中的第 19 天里只有一项活动，因此，若再次在 JSpinner 中选择 19，点击 Pick an event:JComboBox 中的向下箭头，可以看到，有两项活动出现，包含步骤 9 中新添加的活动（如图 25.7 所示）。经过“切换”，确认每次都能在 Description:JTextArea 中显示出正确的活动信息。

- 12. 关闭正在运行的应用程序 点击关闭按钮关闭正在运行的应用程序。
- 13. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。



图 25.6 输入另一项活动并关闭文件

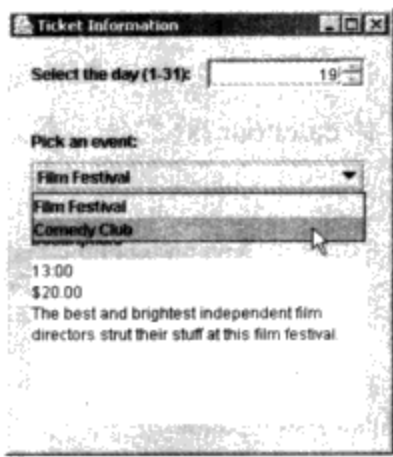


图 25.7 利用 Pick an event:JComboBox 选择在同一天内举行的多项活动

25.2 数据分级

计算机所能处理的所有数据项将构成一个数据分级（如图 25.8 所示）。从位到字符、再到属性，最终变为更大的数据结构，数据项会越来越多，其结构也越来越复杂。

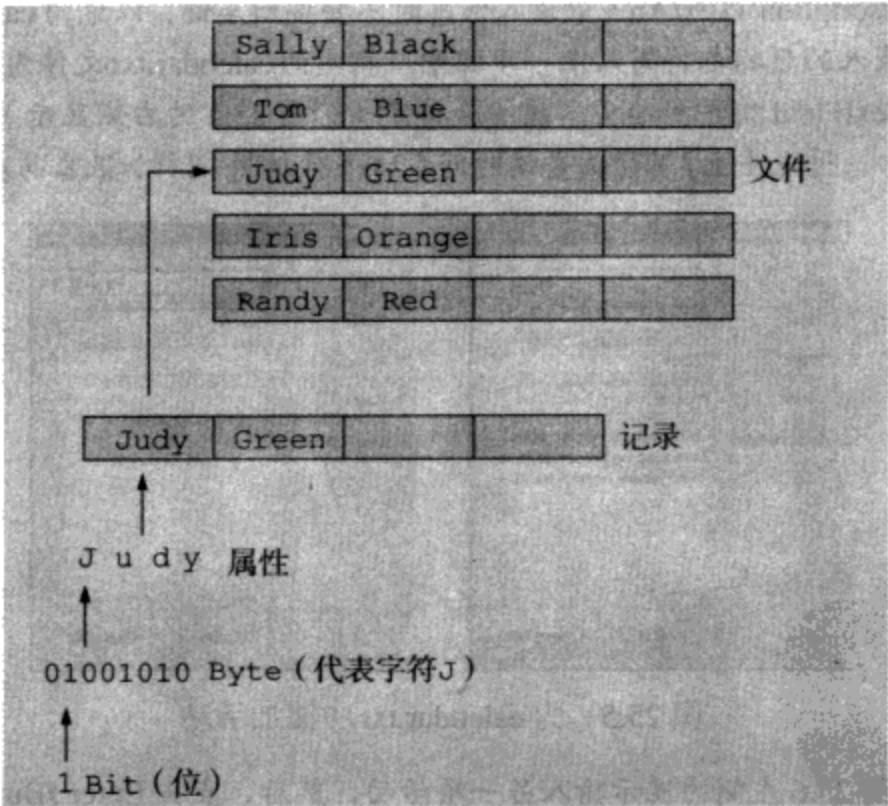


图 25.8 数据分级

自始至终，读者都是通过应用程序实现对数据的处理。数据形式有许多种：十进制数（0，1，2，3，4，5，6，7，8和9）、字母（A~Z和a~z）、一些特殊符号（\$，@，%，&，#，(,)，-，+，"，:，?，/，等等）。数字、字母和特殊符号均被看做字符。用于编写应用程序并表示特定计算机数据项的一整套字符的集合被称为该计算机的字符集。

所有数据项最终都将被计算机转化为0和1的组合。计算机所支持的最小数据项被称为位。“位”是“二进制位”的简称，每一位只可能是两个值中的一个，即0或1。计算机电路能够执行各种简单的位操作，如检查值的大小、设置值的大小、值的翻转（0变1，1变0）。这种方式简单经济，而且设计出具有两种状态的电子设备也相对容易。两种状态，一种状态用0表示，另一种状态用1表示。值得注意的是，计算机所实现的其他扩展功能也仅仅只是涉及最基本0和1的操作。

计算机字符集中的每一个字符也都是一些0和1的组合。一个字节由8位组成。Java采用的字符为Unicode字符，它由两个字节（16位）组成。使用较低级的位的形式来编写数据是很困难的，因此程序员使用字符对应用程序和数据项进行编写，计算机能够操作和处理这些由多个位所组成的字符。

和由位所组成的字符一样，属性则是由字符组成的（如图25.8所示）。每一个属性为一组用于传递某种信息的字符所组成。例如，一个由大写和小写字母组成的属性表示某人的姓名。

记录（Java中用类来表示），通常是多个相关属性的集合（Java中称之为实例变量）。例如，在一个工资单系统中，特定员工的记录可包括以下属性：

1. 员工编号
2. 姓名
3. 地址
4. 每小时的薪资
5. 免税额
6. 年薪
7. 需扣除的税额

所以说，记录代表一组相关属性。在上面的例子中，所有的属性值一起代表某一个员工。

文件则代表一组相关的记录。公司内的工资单文件通常为每一个员工保留一条记录。因此，一个规模不大的公司其工资单文件可能只有22条记录，而一个规模较大的公司则可能会包含100 000条以上的记录。然而，对于一些拥有多个文件，其文件中包含百万、十亿甚至百亿条以上字符信息的公司来说，其实也并不稀奇。

为了能方便地从一个文件中检索出特定的记录，至少应在记录中选择一个属性作为记录关键字。记录关键字用于标识某条记录属于某个特定的个人或团体，并且还可用来区别不同的记录。因此，记录关键字必须是惟一的。在上述工资单记录中，员工编号通常可作为记录关键字，因为每一位员工的编号都是不同的。

文件中记录的组织方式有很多种。最常用的一种方式称为顺序文件，其中，所有记录均按照一个记录关键字属性的顺序进行存储。在工资单文件中，记录通常是以员工编号的顺序来进行存储的。文件中的第一条员工记录为编号最低的一个员工，随后的记录将按照员工编号依次递增的顺序来进行存储。

多数企业使用许多不同的文件来存储数据。例如，某公司会拥有自己的工资单文件、账户收入文件（列有客户金额）、账户支出文件（列有支付给供应商的金额）、库存清单文件（列有该企业经手的所有项目）以及其他类型的一些文件。有时，将一组相关文件称为数据库。用于创建和管理数据库的一组程序则被称为数据库管理系统（DBMS，Database Management System）。读者将在教程26和教程31中学习更多有关数据库的知识。

自测题

1. 计算机能够处理的最小数据项被称为_____。

- a) 数据库 b) 字节 c) 文件 d) 位
- 2. _____ 代表一组相关的记录。
a) 文件 b) 属性 c) 位 d) 字节

答案：1) d 2) a

25.3 文件和流

Java 将文件看做字节或字符的序列，其序列被称为流（如图 25.9 所示）。本教程中，读者将学习如何管理含有字符的文件。Java 应用程序对于文件的处理均使用来自 java.io 包中的类。此包中有许多类，包括用来将字符写入文件及从文件中读取字符的一些类。本节中，将向读者简要介绍其中一些类的使用。

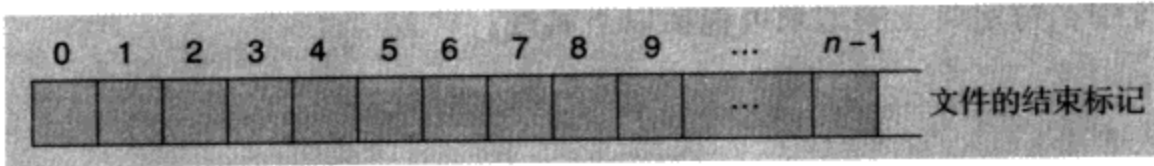


图 25.9 Java 中一个 n 字节文件的概念视图

读者将通过 `FileWriter` 类和 `FileReader` 类的对象打开用于输入和输出字符的文件。这两个类允许向文件中写入字符或者从文件中读出字符。

在对活动录入应用程序的探试过程中，读者了解了该应用程序对一些信息所做的处理，这些信息如字符串和数字，等等。然而，`FileWriter` 对象只能将字符写入文件。在 Java 中，可使用 `PrintWriter` 类的对象将字符串和数字（及其他任何形式）以文本（即字符）的形式进行输出，但 `PrintWrite` 并不知道如何向文件中写入数据。本教程中，读者将学习如何结合 `FileWriter` 和 `PrintWriter` 的功能将应用程序中的字符串和数字输出到一个文本文件中。

类似地，票务信息查询应用程序每次从活动录入应用程序所创建的文件中读取一行文本。然而，`FileReader` 对象只能从文件中读取字符。在 Java 中，可以使用 `BufferedReader` 类的对象一次读取几行，但 `BufferedReader` 并不知道如何从文件中读取数据。读者将学习如何结合 `FileReader` 和 `BufferedReader` 的功能，以行为单位从文件中读取文本。

自测题

- 1. 在与 `FileWriter` 类结合使用时，_____ 类能够将字符串和数值写入一个文件。
a) `OutputFile` b) `StreamWriter` c) `PrintWriter` d) `BufferedReader`
- 2. Java 将文件看做为一个字节 _____ 序列。
a) 流 b) 循环 c) 字符串 d) 记录

答案：1) c 2) a

25.4 创建活动录入应用程序：向文件中写入信息

下面，读者创建活动录入应用程序，用户将把社团的活动信息写入一个按顺序存取的文件中。首先，我们需要对该应用程序进行分析。下列伪代码描述了活动录入应用程序的基本操作。

当用户点击 `Open File...JButton` 时
 显示 `JFileChooser` 对话框
 返回用户所选择的文件

打开用于写入信息的此文件
禁用 Open File...JButton
启用 Enter JButton
启用 Close File JButton
重置输入组件

当用户点击 Enter JButton 时
向文件中添加活动日期及换行符
向文件中添加活动时间及换行符
向文件中添加活动价目及换行符
向文件中添加活动名称及换行符
向文件中添加活动描述及换行符
重置输入组件

当用户点击 Close File JButton 时
关闭文件
禁用 Enter JButton
启用 Open File...JButton
禁用 Close File JButton
重置输入组件

既然读者已探试了活动录入应用程序并研究了它的伪代码表示,下面,我们将使用一张 ACE 表,帮助读者把这个伪代码转换成 Java 实现。图 25.10 中列出了该应用程序中相应的操作、组件及事件,帮助读者最终完成属于自己的这一应用程序。

操作	组件 / 对象	事件
 标记应用程序中的组件	dayJLabel, timeJLabel, priceJLabel, eventJLabel, descriptionJLabel	当运行应用程序时
显示 JFileChooser 对话框	openFileJButton fileChooser (JFileChooser)	当用户点击 Open File ...JButton 时
返回用户选择的文件	fileChooser, selectedFile(File)	
打开用于写入信息的此文件	outputFile(FileWriter), output(PrintWriter)	
禁用 Open File...JButton 启用 Enter JButton 启用 Close File JButton 重置输入组件	openFileJButton enterJButton closeFileJButton priceJTextField, eventJTextField, descriptionJTextArea	
向文件中添加活动日期及换行符	enterJButton	当用户点击 Enter JButton 时
向文件中添加活动时间及换行符	dayJSpinner, output(PrintWriter)	
向文件中添加活动价目及换行符	timeJSpinner, output(PrintWriter)	
向文件中添加活动名称及换行符	priceJTextField output(PrintWriter)	
向文件中添加活动描述及换行符	eventJTextField, output(PrintWriter)	
重置输入组件	descriptionJTextArea, output(PrintWriter) priceJTextField, eventJTextField, descriptionJTextArea closeFileJButton	当用户点击 Close File JButton

(续表)

操作	组件/对象	事件
关闭文件	output(PrintWriter)	
禁用 Enter JButton	enterJButton	
启用 Open File...JButton	openFileJButton	
禁用 Close File JButton	closeFileJButton	
重置输入组件	priceJTextField, eventJTextFiled, descriptionJTextArea	

图 25.10 活动录入应用程序的 ACE 表

票务信息查询应用程序的一个重要特征是能够从文件中按顺序读出数据。读者需要创建一个供票务信息查询应用程序读取数据的文件。因此，在创建票务信息查询应用程序之前，必须首先学习如何向文件中按序写入数据。活动录入应用程序将把用户输入的信息存储在一个文本文件中。用户通过提供的输入组件输入某项活动的日期、时间、名称以及相关的信息描述。利用一些 JButton 实现文件的打开、写入及关闭操作。所有数据均通过输入组件被录入，然后写入一个由用户指定的文件中。为实现文本文件的打开及写入操作，需要在应用程序中分别创建一个 FileWriter 对象和一个 PrintWriter 对象。活动录入应用程序可让用户创建一个新的文件或打开一个已有文件，如果已有文件已被打开，则新的活动项目将被添加到该文件的末尾。

创建一个 PrintWriter 对象

- 1. 将模板复制到工作目录中 将 C:\Examples\Tutorial25\TemplateApplication\WriteEvent 目录复制到 C:\SimplyJava 目录中。
- 2. 打开活动录入应用程序的模板文件 在自己的文本编辑器中打开模板文件 WriteEvent.java。
- 3. 导入处理文件的 java.io 包 将图 25.11 中第 6 行添加到代码中，该代码行将导入一个 java.io 包。java.io 包为用户提供了执行按顺序存取文件处理时所需要的类及其方法。

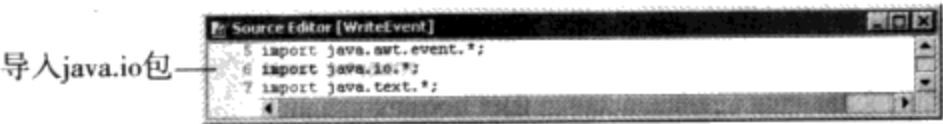


图 25.11 把 java.io 包导入 WriteEvent 类中

- 4. 声明一个 PrintWriter 对象 将图 25.12 中第 39 行和第 40 行添加到程序代码中，声明了一个 PrintWriter 型实例变量 output。在本教程后面，读者将使用这一变量向文件中写入数据。

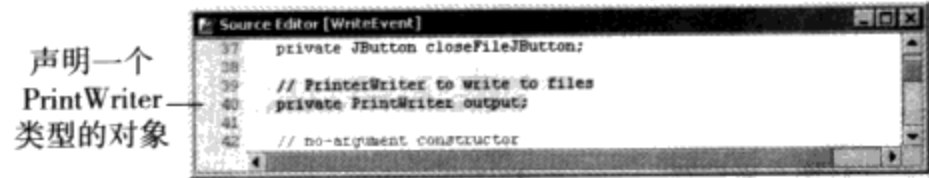


图 25.12 声明一个 PrintWriter 类型的对象

- 5. 显示 JFileChooser 对话框 将图 25.13 中第 186 行和第 189 行插入到 openFileJButtonActionPerformed 方法中。第 187 行通过创建一个 JFileChooser 对象，允许用户打开一个文件。JFileChooser 对象用于显示一个让用户从磁盘中选择某个文件的对话框。在应用程序中，如果用户指定的文件不存在，则会按照指定名创建一个新的文件。第 188 行通过调用 JFileChooser 的 setDialogTitle 方法设置 JFileChooser 对话框标题栏内显示的字符串。第 189 行通过调用 showOpenDialog 方法显示了一个 JFileChooser 对话框。showOpenDialog 方法需接收一个父 GUI 组件 (this) 作为参数，表示对话框将居中于应用程序主窗口之上。另外，该方法返回的一个 int 值表明用户是否点击了对话框中的 OK JButton 或 Cancel JButton。例如，当用户点击 Cancel JButton 时，showOpenDialog 方法返回的一个 int 型常量为 JFileChooser.

CANCEL_OPTION; 当点击 OK JButton 时, 返回一个 int 型常量 JFileChooser.APPROVE_OPTION。如果有错误发生, 则返回一个 int 型的常量 JFileChooser.ERROR_OPTION。

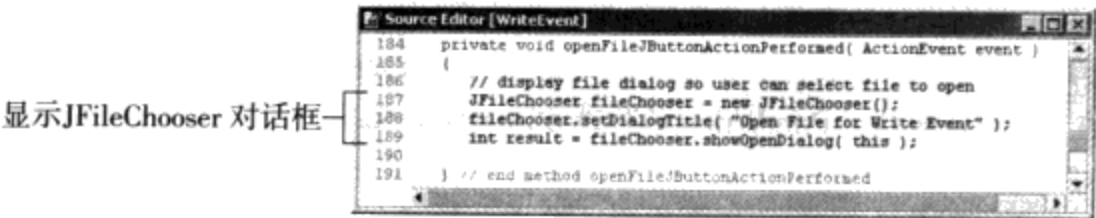


图 25.13 显示 JFileChooser 对话框并返回结果

6. 若点击 Cancel JButton 退出方法 将图 25.14 中第 191 行至第 195 行添加到应用程序中。如果用户点击 Cancel JButton (参见第 192 行), 位于第 194 行上的 return 语句会让该方法退出执行, 这样, 用户可重新点击 Open File…JButton 并选择一个要打开的文件。

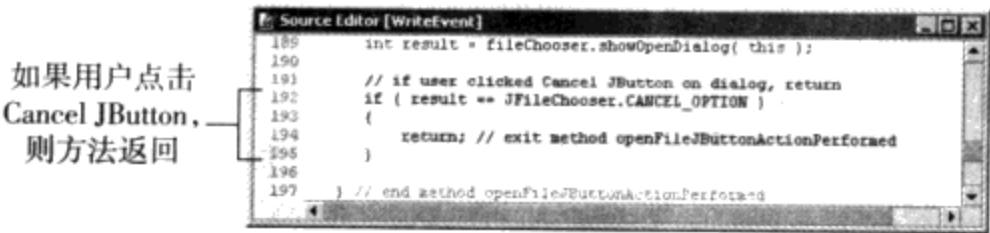


图 25.14 如果用户点击 Cancel JButton, 则方法退出执行

7. 返回文件 将图 25.15 中第 197 行至第 198 行添加到程序代码中。第 198 行通过调用 JFileChooser 的 getSelectedFile 方法返回用户所选择的文件, 返回的文件为一个 File 类的对象。对于 File 类, 正如读者将在下一步骤中看到的那样, 用来返回一个与文件或目录相关的信息。

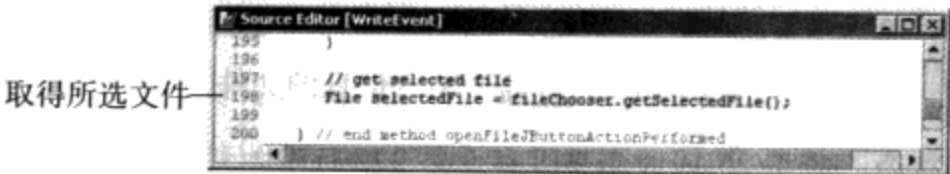


图 25.15 返回用户选择的文件

8. 获取所选文件的文件名 将图 25.16 中第 200 行至第 201 行添加到程序代码中。File 类的 getName 方法能够将所选文件 (参见第 201 行) 的名称返回为一个字符串。

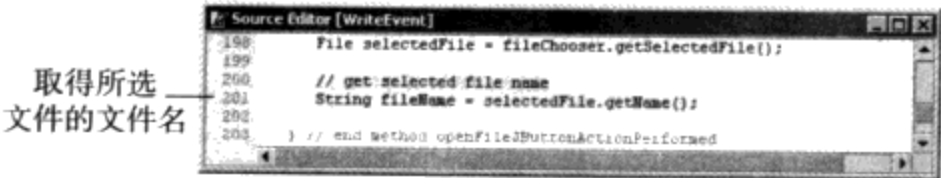


图 25.16 通过 getName 方法返回文件名

9. 检查文件名是否存在 将图 25.17 中第 203 行至第 208 行添加到程序代码中。第 204 行用来判定文件名是否存在。如果文件不存在, 通过第 206 行至第 207 行的代码显示一个错误消息对话框。

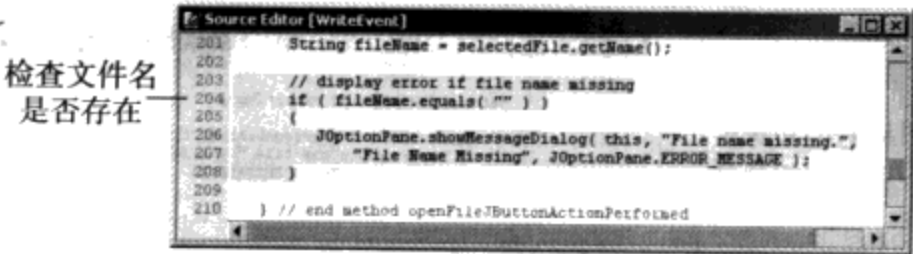


图 25.17 确认文件名是否有效

10. 初始化一个 PrintWriter 对象 将图 25.18 中第 209 行至第 220 行添加到程序代码中。第 215 行至第 216 行通过创建一个 FileWriter 对象将字符写入一个由 selectedFile 指定的文件中 (作为 FileWriter 构造方法中

的第一个参数)。前面曾经讲过，selectedFile 代表用户通过 JFileChooser 对话框选择的一个文件。第二个参数表明新的字符将添加到文件中的位置。如果此参数值为 true (如第 216 行所示)，任何新写入文件中的数据都将被追加到文件的末尾；相反，如果参数值为 false，则数据将写入文件的起始位置处并覆盖原先已存储的数据。一旦第 215 行和第 216 行创建完 FileWriter，便可执行文件的写入操作。第 217 行将 FileWriter 作为参数传递给了 PrintWriter 的构造方法并初始化了一个 PrintWriter 的对象 output，此对象最终将把文本写入由 fileName 指定的文件中。

注意，该段代码放置在了一个 try 语句块中，因为在打开文件的时候如果出现问题（例如，打开文件时，所在磁盘的空间不足），FileWriter 的构造方法可能会抛出一个 IOException 异常（表示输入/输出错误）。此异常将在 catch 语句块中进行处理。



软件设计提示

当调用 FileWriter 的构造方法打开一个已有文件时，若第二个参数为 false，则文件中以前保留的数据会发生丢失。



常见编程错误

试图处理一个未经打开的文件会产生 IOException 异常，该异常表示输入/输出发生错误。

创建FileWriter并将其传递
给PrintWriter的构造方法

```
Source Editor [WriteEvent *]
208     }
209     else
210     {
211         // open file
212         try
213         {
214             // open file for writing
215             FileWriter outputFile =
216                 new FileWriter( selectedFile, true );
217             output = new PrintWriter( outputFile );
218         }
219     } // end else
220
221 } // end method openFile/ButtonActionPerformed
```

图 25.18 打开需要写入信息的文件

11. 改变 JButton 的状态 将图 25.19 中第 219 行至第 222 行添加到程序代码中以改变 JButton 的状态。第 220 行用于防止用户在未关闭当前文件之前就去打开另外一个文件。第 221 行至第 222 行实现用户在文件中添加数据以及关闭文件的操作。

```
Source Editor [WriteEvent]
217     output = new PrintWriter( outputFile, true );
218
219     // change state of JButtons
220     openFileJButton.setEnabled( false );
221     enterJButton.setEnabled( true );
222     closeFileJButton.setEnabled( true );
223 }
224
225 } // end else
```

图 25.19 改变 JButton 的状态

12. 捕获 IOException 异常 将图 25.20 中第 224 行至第 229 行添加到应用程序代码中，作为捕获可能从第 212 行至第 223 行中的 try 语句块中抛出的 IOException 异常。

捕获任何从try语句块中
抛出的IOException异常

```
Source Editor [WriteEvent]
223     }
224     catch ( IOException exception )
225     {
226         JOptionPane.showMessageDialog( this,
227             "Cannot open the file " + fileName + ". Error",
228             JOptionPane.ERROR_MESSAGE );
229     }
230
231 } // end else
```

图 25.20 若捕获 IOException 异常，则显示一个 JOptionPane

13. 重置输入组件 将图 25.21 中第 233 行至第 234 行添加到程序代码中。第 234 行通过调用 resetUserInput 方法（由模板提供），实现对 priceJTextField，eventJTextField 和 descriptionJTextArea 中文本的清除。

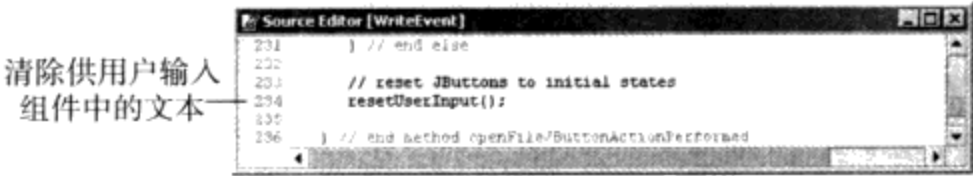


图 25.21 调用 resetUserInput 方法

- 14. 保存应用程序 保存修改后的源代码文件。
- 15. 编译应用程序 通过键入 `javac WriteEvent.java` 编译该应用程序。
- 16. 运行应用程序 若能正确编译应用程序，键入 `java WriteEvent` 来运行它。图 25.22 显示了更新后的应用程序的运行结果。Time:JSpinner 中显示为当前时间（8 PM）。
- 17. 创建一个文件 点击 Open File...JButton，出现 Open File for Write Event 对话框。浏览至 C:\SimplyJava\WriteEvent 目录并在 File Name:field 中输入 test.txt，点击 Open JButton。以上操作将创建一个 test.txt 文件并存储在 WriteEvent 目录中。

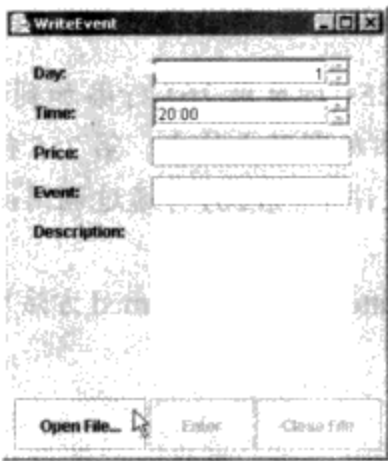


图 25.22 用于打开文件的活动录入应用程序

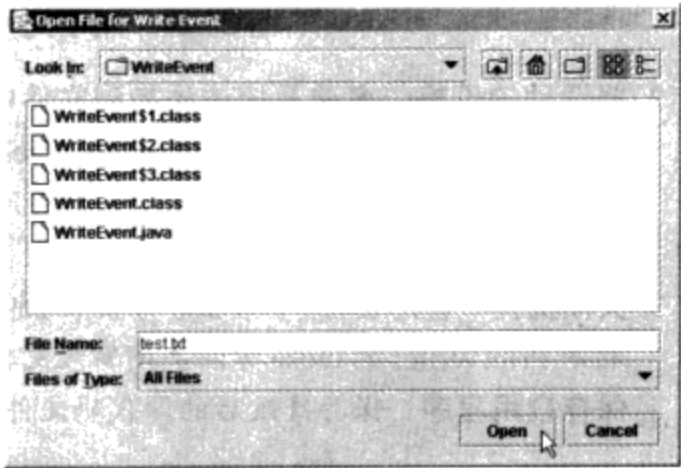



图 25.23 创建一个文件

上述操作结束以后，将返回到应用程序的窗口中。虽然 Enter 和 Close File JButton 已经启用，但仍未向其添加功能，因此，点击时不会执行任何操作。关闭应用程序，浏览 WriterEvent 目录，注意新创建了一个 test.txt 文件，但该文件内并无任何数据。同样，读者也可使用当前应用程序打开一个已有文件，但只有在下一部分学完如何向文件中写入数据以后，才能实现如何向其添加数据。

- 18. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- 19. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。



好的编程习惯

一旦应用程序不再需要某个文件，就应立即关闭该文件，从而最大程度地降低破坏文件的可能性，并允许其他应用程序对该文件进行访问。

此应用程序现在已能够打开一个文件。下面，学习如何将输入的信息写入文件中。读者将向 enterJButtonActionPerformed 方法添加相应的代码，用于执行点击 Enter JButton 时的操作。此方法将把用户输入的数据写入一个文本文件中。读者还将添加当点击 Close File JButton 时，关闭文件的代码。当应用程序不再需要某个文件时，应立即关闭该文件，从而最大程度地降低对文件破坏的可能性并允许其他应用程序访问该文件。

向一个按顺序存取的文件中写入信息

- 1. 向文件中写入某项活动的日期和时间 将图 25.24 中第 241 行至第 249 行添加到 enterJButtonActionPerformed 方法中。第 242 行至第 249 行利用 PrintWriter 的 println 方法将用户输入的信息逐行写入一个

文件。此方法可接收基本类型或 Object 对象作为参数并使用参数值的 String 形式写入文件，之后会跟一个换行符。第 242 行的 println 方法把用户输入的日期写入到一个文件中，后跟一个换行符，即下起新的一行。读者将把每一条信息添加到文件中独立的一个行上。第 245 行用于获取用户输入的时间。timeJSpinner.getValue 所返回的数据实际多于需要的数据，比如包含了当前的月份。对于该组件，我们关心的只是活动举办的小时和分钟数据。第 246 行利用 substring 方法可提取出这一相关信息。第 249 行把时间信息写入到文件中并进行换行。

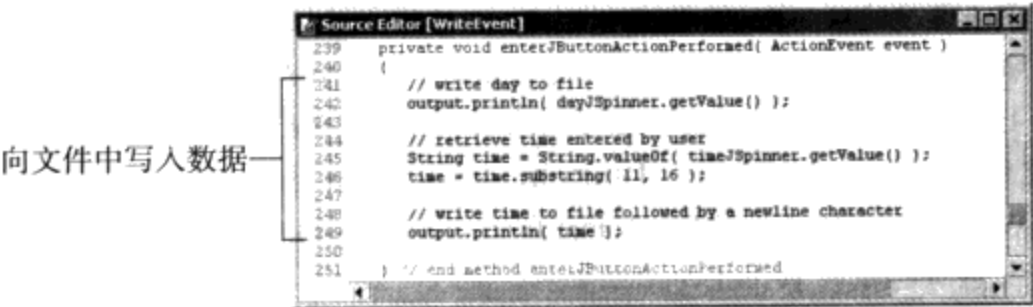


图 25.24 用于向文件中写入信息的 PrintWriter

- 2. 将活动的价格、名称及描述添加到文件中 将图 25.25 中第 251 行至第 261 行添加到程序代码中。第 252 行把一个美元符号和 priceJTextField 中输入的价格数据添加到了文件中。第 255 行是将此项活动的名称添加到文件中。第 258 行则是将活动的描述添加到了文件中。第 261 行通过调用 resetUserInput 方法清除 JTextField 输入组件内的文本。
- 3. 关闭文件 将图 25.26 中第 268 行至第 269 行添加到 closeFileJButtonActionPerformed 方法中。第 269 行利用 PrintWriter 的 close 方法关闭写入文件的流。
- 4. 保存应用程序 保存修改后的源代码文件。

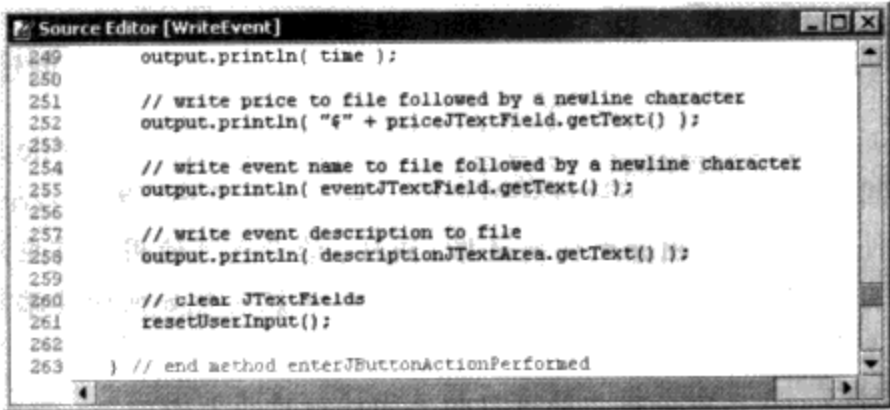


图 25.25 向文件中添加活动的价格、名称以及描述信息

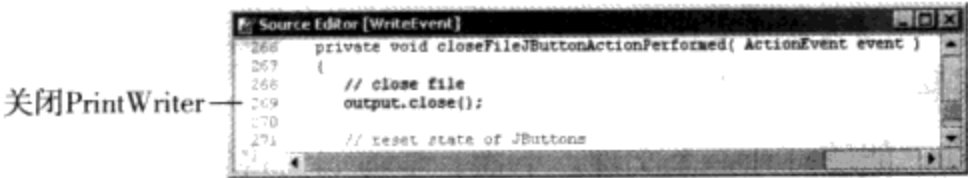


图 25.26 关闭 PrintWriter

- 5. 编译应用程序 键入 javac WriteEvent.java 编译该应用程序。
- 6. 运行应用程序 若能正确编译应用程序，键入 java WriteEvent 来运行它。图 25.27 显示了更新后的应用程序的运行结果。
- 7. 创建或打开一个文件 点击 Open File...JButton，创建或打开一个用于写入数据的文件。这时，出现一个 Open 对话框（如图 25.28 所示）。浏览到 C:\SimplyJava\WriteEvent 目录并打开 test.txt 文件。
- 8. 输入活动信息 在 Day:JSpinner 中选择 4，表示该项活动将安排在本月第 4 天。在 Time:JSpinner 中输入 14: 30，在 Price:JTextField 中输入 12.50，在 Event:JTextField 中输入 Arts 和 Crafts Fair，最后，再向 Description:JTextArea 中输入以下信息: Take part in creating various types of arts and crafts at this fair。点击 Enter JButton 将该项活动的信息添加到文件 test.txt 中。

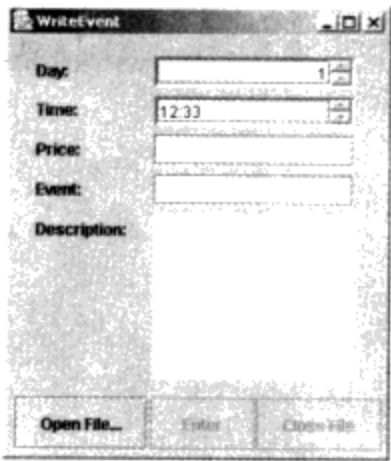


图 25.27 活动录入应用程序的运行结果

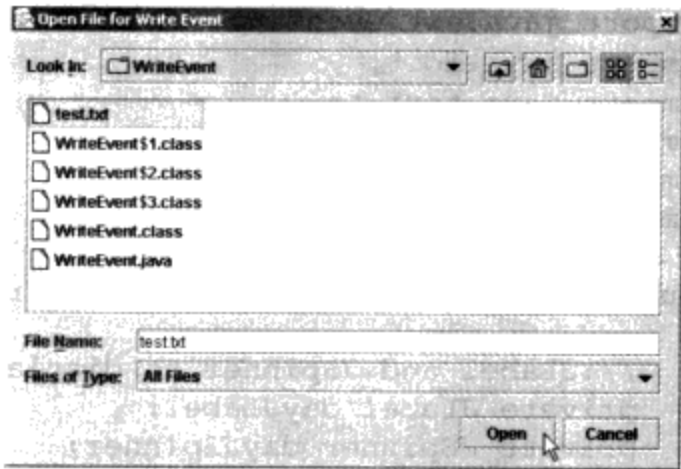


图 25.28 显示活动录入应用程序模板目录内容的 Open File for Write Event 对话框

- 9. 关闭文件 输入完所有活动以后，点击 Close File JButton，关闭 test.txt 文件，防止其被改写（直到打开另一个文件或再次打开 test.txt 文件）。关闭应用程序之前应记得点击 Close File JButton。
- 10. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- 11. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。
- 12. 打开和关闭按顺序存取的文件 利用文本编辑器中打开 test.txt 文件。滚动至文件的末尾。步骤 8 中所输入的信息应出现在文件中，如图 25.30 所示。正常情况下，信息中的每一条数据并不是以这种方式（独立一行）来存储的。相反，每一项活动的所有信息都应存放在同一行内，各部分之间由一个特殊字符分隔（如空格或 tab 键）。这里，我们把文件按照如图 25.30 的形式进行组织只是为简单方便，对于每一条信息的读取可通过读取文件中的一行来实现，而不必一次读取所有信息，然后再通过它们之间的特殊字符进行分离。完成以后，关闭 test.txt 文件。



图 25.29 将 Arts 和 Crafts Fair 添加到 test.txt 文件中

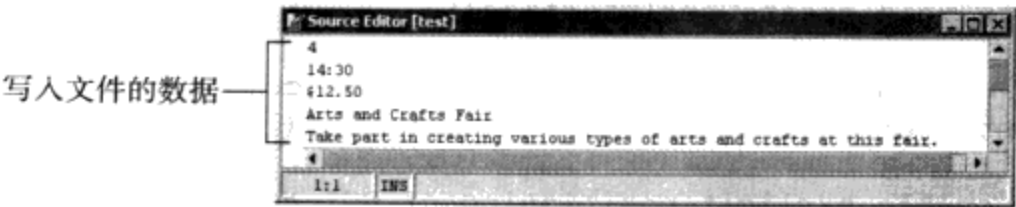


图 25.30 活动录入应用程序所产生的按顺序存取文件

图 25.31 列出了活动录入应用程序的源代码。在本教程中，凡需要添加、查看或是修改的代码均在图中相应的代码行中进行了突出显示。

```
1 // Tutorial 25: WriteEvent.java
2 // This application writes information about an event on a given
3 // date to a file.
4 import java.awt.*;
```

自测题

1. min 方法和 abs 方法都属于 _____ 类。
a) Calc b) Math c) Calculation d) Number
2. drawLine, fillOval 和 setColor 方法都属于 _____ 类。
a) Draw b) Graphics c) Drawing d) Graphic

答案: 1) b 2) b

27.5 小结

在本教程中,学习了有关多态的概念。通过绘图应用程序,能够将许多不同颜色的图形组合成一幅图画。同时,还学习了如何使用 Graphics 的不同方法绘制直线、填充矩形及填充椭圆。

在构建绘图应用程序的过程中,读者接触了一个由超类 MyShape 及其子类 MyLine, MyRectangle, MyOval 所构成的继承结构。与此同时,根据多态将它们看做是超类 MyShape 的对象,并对这三个子类的对象进行了相应处理。

在下一个教程中,将学习 Java 语音 API 的知识,为输入的文本产生合成的语音。读者将使用这一技术创建一个电话簿应用程序,该应用程序能够读出所选人的电话号码。

技术小结

绘制矩形

- 使用 Graphics 类的 drawRect 方法绘制一个由 x 坐标、 y 坐标、宽度和高度所指定的矩形。

绘制椭圆

- 使用 Graphics 类的 drawOval 方法绘制一个椭圆,该椭圆内切于一个由 x 坐标、 y 坐标、宽度和高度所指定的矩形框。

绘制直线

- 使用 Graphics 类的 drawLine 方法绘制一个由起点和终点的 x 坐标、 y 坐标所指定的直线。

关键术语

Math 类的 abs 方法 返回某值的绝对值。

绝对值 不带符号的数值。

抽象类 一种不能被实例化的类。通常称为抽象超类,因为只有在继承结构中超类才是有意义的。抽象类是不完整的类,缺失的部分必须在具体子类中实现。

abstract 关键字 定义抽象类或抽象方法的关键字。

抽象方法 只有方法头而无方法体的方法。任何含抽象方法的类必为抽象类。

具体类 能够被实例化的类。

Graphics 类的 drawLine 方法 利用给定的 x 坐标和 y 坐标绘制一条直线。

Graphics 类的 drawRect 方法 利用给定的 x 坐标和 y 坐标、宽度和高度绘制一个矩形。

Math 类的 min 方法 返回两个值中的较小值。

多态 以更为通用的方式及大量具有继承关系的类来编写应用程序的一个概念。

Java 类库索引

Graphics Graphics 类中提供了许多绘制不同颜色图形的方法。

● 方法

drawLine 绘制一条直线,接收 4 个参数,分别为起点和终端点的 x 坐标和 y 坐标。

drawOval 绘制一个内切于外框矩形的非填充椭圆。接收 4 个参数, 前两个参数为矩形框左上角的 x 坐标和 y 坐标, 后两个参数为矩形框的宽度和高度。

drawRect 绘制一个实心矩形。接收 4 个依次代表矩形左上角的 x 坐标、 y 坐标、宽度和高度的参数。

fillOval 绘制一个内切于矩形框的实心椭圆。接收 4 个参数, 前两个参数为矩形框左上角的 x 坐标和 y 坐标, 后两个参数为矩形框的宽度和高度。

setColor 设置 Graphics 对象的颜色。

Math Math 类中提供了许多实现数学函数计算的方法。

● 方法

abs 返回参数值的绝对值。

max 返回两个参数值的较大值。

min 返回两个参数值的较小值。

习题

选择题

- 27.1 代码 _____ 能够绘制一个实心圆。
- a) drawCircle(50, 50, 25) b) fillOval(50, 25, 50, 25)
c) fillOval(50, 50, 25, 25) d) drawOval(50, 50, 50, 50)
- 27.2 利用多态, 使用同一个 _____ 可产生不同的操作, 这取决于调用方法的对象类型。
- a) 方法返回类型 b) 实例变量 c) 局部变量 d) 方法名
- 27.3 方法 _____ 能够返回某个数的绝对值。
- a) abs b) absolute c) positive d) positiveValue
- 27.4 如果 MyTruck 类继承 MyCar 类, 则 _____。
- a) MyTruck 类的对象可以赋值给 MyCar 类型的变量
b) MyCar 类的对象可以赋值给 MyTruck 类型的变量
c) 两个类的对象可以相互赋值给对方类型的变量
d) (a)和(b)
- 27.5 多态能够让开发人员 _____ 进行编程。
- a) “以抽象的形式” b) “以全局的形式” c) “以特定的形式” d) (a)和(b)
- 27.6 drawLine 方法中的第一参数和第二个参数代表直线的 _____ 坐标。
- a) 左上角 b) x c) y d) 以上答案都不对
- 27.7 诸如 drawOval 和 drawRect 的方法定义在 _____ 类中。
- a) Drawing b) Paint c) Graphics d) Images
- 27.8 对于使用多态的应用程序来说, 某个对象的确切类型 _____。
- a) 执行时才可以知道 b) 应用程序编译时能够判断
c) 程序员编写程序时就已知道 d) 永远不会知道
- 27.9 _____ 能够绘制一条垂直直线。
- a) drawLine(50, 50, 25, 25) b) drawLine(25, 25, 50, 25)
c) drawLine(50, 25, 50, 25) d) drawLine(50, 25, 50, 50)
- 27.10 通过多态, 使用 _____ 类型的变量可以调用超类和子类对象的方法。
- a) 基本 b) 超类 c) 子类 d) 以上答案都不对

练习题

- 27.11 (高级屏保应用程序) 编写一个应用程序, 模仿屏保程序的运行过程。该应用程序能够不断地将随机图形绘制于一个黑色背景之上, 且新绘制的图形会盖过其他图形, 整个过程重复执行直到屏保应用程序重新复位 (每隔 30 秒执行一次复位操作)。这里, 已为读者提供了屏保应用程序, 但

该应用程序尚未实现绘图功能。此屏保应用程序将使用本教程中创建的MyRectangle类和MyOval类。读者需要为其添加绘制随机图形的代码，完成后的应用程序的运行结果如图 27.35 所示。

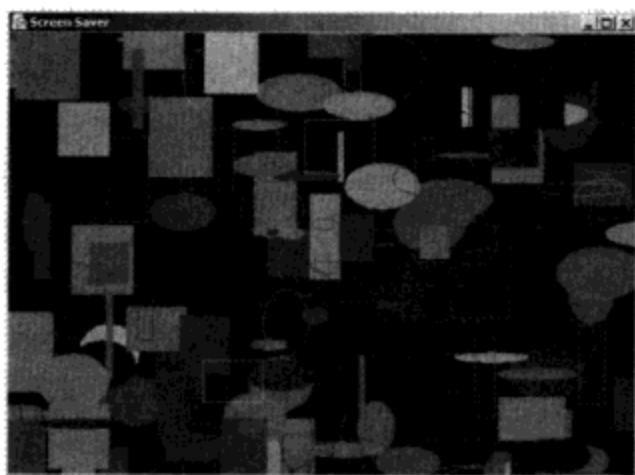


图 27.35 高级屏保应用程序

- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial27\Exercises\AdvancedScreenSaver 目录复制到 C:\SimplyJava 目录中。
- b) 打开模板文件 在自己的文本编辑器中打开 MyRectangle.java 文件。
- c) 在 MyRectangle 类中添加一个实例变量 在第 7 行，加入一段注释，说明下面的实例变量是一个 boolean 型变量，代表矩形是否为填充图。在第 8 行，添加一个类型为 boolean 的 private 实例变量 filled。
- d) 修改 MyRectangle 类的构造方法 接下来，修改 MyRectangle 类的构造方法，使它能够再接收一个 boolean 型参数。在第 12 行，将一个名为 fill 的 boolean 型参数添加到参数列表的末尾。在第 16 行，将实例变量 filled 设置为参数 fill 的值，然后添加一行注释，说明 filled 的值代表该图形是否为填充图。
- e) 修改 draw 方法 在第 31 行，加入一行注释，表明以下 if 语句会在矩形为填充图时开始执行。在第 32 行，添加一条 if 语句，检查 filled 是否为 true。如果成立，则调用 fillRect 方法（位于模板中第 30 行）。
- f) 完成 draw 方法 在第 37 行，为上一步中完成的 if 语句添加一条 else 语句。如果 filled 为 false，则调用 drawRect 方法。
- g) 保存应用程序 保存修改后的源代码文件。
- h) 打开模板文件 在自己的文本编辑器中打开 MyOval.java 文件。
- i) 修改 MyOval 类 将上述步骤(c)至(f)应用到 MyOval 类中。MyOval 类的行号与 MyRectangle 类相同，分别使用 fillOval 和 drawOval 方法取代 fillRect 和 drawRect 方法。
- j) 保存应用程序 保存修改后的源代码文件。
- k) 打开模板文件 在自己的文本编辑器中打开 DrawJPanel.java 文件。
- l) 修改针对不同图形构造方法的调用 下面，将把一个 boolean 参数添加到图形构造方法的调用语句中。在第 117 行，在参数列表的末尾添加一个参数。修改后的语句能够创建一个椭圆轮廓，即非填充椭圆。因此，添加的参数值应为 false。这会使 MyOval 类的实例变量 filled 被设置为 false。在第 123 行，在参数列表的末尾添加一个 true 值。此时，当执行该行代码时，会创建一个 MyOval 对象，其实例变量 filled 的值为 true。在第 130 行，在参数列表的末尾添加一个参数值 false。当执行该行代码时，会创建一个 MyRectangle 对象，其实例变量 filled 的值为 false。最后，在第 136 行，在参数列表的末尾添加一个 true 值。当执行该行代码时，会创建一个 MyRectangle 对象，其实例变量 filled 的值为 true。
- m) 保存应用程序 保存修改后的源代码文件。
- n) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\AdvancedScreenSaver 进入到当前工作目录中。

- o) 编译应用程序 键入 `javac ScreenSaver.java DrawJPanel.java MyRectangle.java MyOval.java` 编译应用程序。
 - p) 运行完成后的应用程序 若能正确编译应用程序, 键入 `java ScreenSaver` 来运行它。在测试应用程序的过程中, 确保能够出现各种图形, 并且每隔 30 秒屏幕自动执行一次清除操作。
 - q) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
 - r) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。
- 27.12 (徽标设计器应用程序) 编写一个应用程序, 设计公司徽标。该应用程序将使用一个简单的坐标输入界面, 绘制直线及填充或非填充的矩形和椭圆, 其 GUI 如图 27.36 所示。

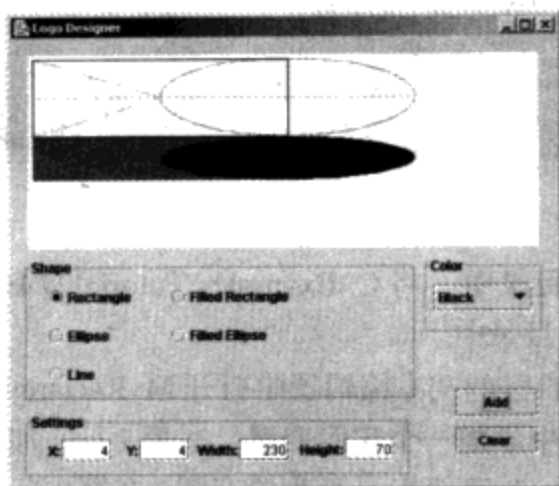


图 27.36 徽标设计器应用程序的 GUI

- a) 将模板复制到工作目录中 将 `C:\Examples\Tutorial27\Exercises\LogoDesigner` 目录复制到 `C:\SimplyJava` 目录中。
- b) 打开模板文件 在自己的文本编辑器中打开 `MyRectangle.java` 和 `MyOval.java` 文件。
- c) 修改 `MyRectangle` 和 `MyOval` 类 将上一练习中的步骤(c)至(j)应用到 `MyRectangle` 类和 `MyOval` 类中, 实现填充或非填充图形的绘制。
- d) 打开模板文件 在自己的文本编辑器中打开 `DrawJPanel.java` 文件。
- e) 添加 `addShape` 方法 在第 31 行, 加入一行注释, 说明以下方法将添加到 `shapeArray` 中并完成图形的刷新功能。在第 32 行, 添加 `addShape` 方法的方法头。此方法接收一个名为 `shape` 的 `MyShape` 型参数, 且方法无返回值。通过调用 `shapeArrayList` 的 `add` 方法并为其传递参数 `shape`, 将 `shape` 加入到 `shapeArrayList` 中。之后, 调用 `repaint` 方法, 使新加入的图形能够显示出来。确保此方法的右花括号结束于第 37 行。
- f) 保存应用程序 保存修改后的源代码文件。
- g) 打开模板文件 在自己的文本编辑器中打开 `LogoDesigner.java` 文件。
- h) 调用绘制直线的 `addShape` 方法 为了在 `JPanel` 上绘制一条新的直线, 需调用 `addShape` 方法。在第 279 行至第 280 行, 调用变量 `drawingJPanel` 的 `addShape` 方法并为其传递参数 `x`, `y`, `width`, `height` 和 `drawColor`, 从而创建一个新的 `MyLine` 对象。
- i) 调用绘制椭圆的 `addShape` 方法 为了在 `JPanel` 上绘制一个椭圆轮廓, 调用 `addShape` 方法。在第 284 行至第 285 行, 调用变量 `drawingJPanel` 的 `addShape` 方法并为其传递参数 `x`, `y`, `x + width`, `y + height`, `drawColor` 和 `false`, 从而创建一个新的 `MyOval` 对象。在第 289 行至第 290 行, 再次调用 `addShape` 方法, 通过将参数列表末尾处的 `boolean` 值改为 `true`, 绘制一个填充椭圆 (非椭圆轮廓)。
- j) 调用绘制矩形的 `addShape` 方法 为了在 `JPanel` 上绘制一个矩形轮廓, 调用 `addShape` 方法。在第 294 行至第 295 行, 调用变量 `drawingJPanel` 的 `addShape` 方法并为其传递参数 `x`, `y`, `x + width`, `y + height`, `drawColor` 和 `false`, 从而创建一个新的 `MyRectangle` 对象。在第 299 行至第 300 行, 再次调用 `addShape` 方法, 通过将参数列表末尾处的 `boolean` 值改为 `true`, 绘制一个填充矩形 (非矩形轮廓)。

- k) **保存应用程序** 保存修改后的源代码文件。
 - l) **打开命令提示符窗口改变目录** 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\LogoDesigner` 进入到当前工作目录中。
 - m) **编译应用程序** 通过键入 `javac LogoDesigner.java DrawJPanel.java MyRectangle.java MyOval.java` 编译该应用程序。
 - n) **运行完成后的应用程序** 若能正确编译应用程序, 键入 `java LogoDesigner` 来运行它。绘制具有不同 x 坐标、 y 坐标、高度和宽度的图形, 完成对应用程序的测试。
 - o) **关闭应用程序** 点击关闭按钮关闭正在运行的应用程序。
 - p) **关闭命令提示符窗口** 点击命令提示符窗口的关闭按钮将其关闭。
- 27.13 (棒打鼹鼠应用程序) 创建一个棒打鼹鼠^①游戏应用程序, 用户点击某个按钮可以开始新的游戏, 之后, 一个鼹鼠会随机出现在预先划定好的单元格内。鼹鼠移动之前如果能击中它, 便增加50分。游戏完成后的结果如图 27.37 所示。

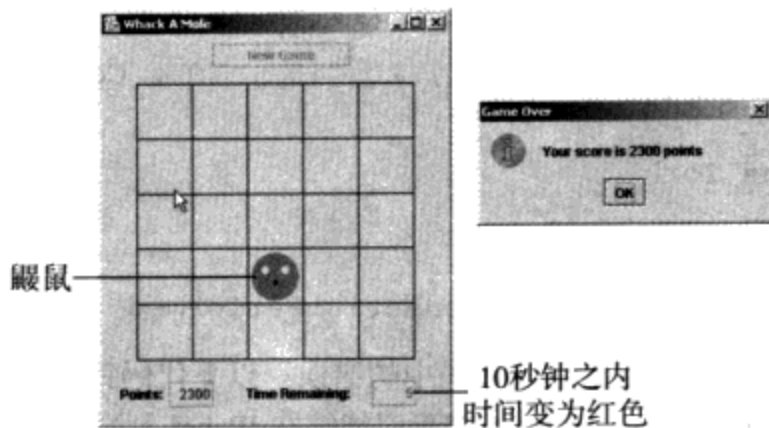


图 27.37 棒打鼹鼠应用程序的 GUI

- a) **将模板复制到工作目录中** 将 `C:\Examples\Tutorial27\Exercises\WhackAMole` 目录复制到 `C:\SimplyJava` 目录中。
- b) **打开模板文件** 在自己的文本编辑器中打开 `Mole.java` 文件。
- c) **在 `drawMole` 方法中声明局部变量** 在第 23 行, 加入一行注释, 说明以下将计算单元格的位置。在第 24 行, 定义并初始化一个 `int` 型局部变量 x , 将 x 设置为 `moleColumn * 50`。接下来, 定义并初始化另外一个 `int` 型局部变量 y , 将 y 设置为 `moleRow * 50`。变量 x 和 y 代表单元格的 x 坐标和 y 坐标。它们将在随后的计算中被用到。
- d) **在 `drawMole` 方法中绘制鼹鼠头** 在第 27 行, 加入一行注释, 说明以下将设置鼹鼠头的颜色。此时, 注意 `drawMole` 方法的参数列表中有一个 `Graphics` 类的实例 g 。在第 28 行, 调用 g 的 `setColor` 方法并为其传递一个 `Color` 对象。 `Color` 构造方法中的整数值为 155, 126 和 87。接下来, 加入一行注释, 说明开始绘制鼹鼠头。调用 g 的 `fillOval` 方法并为其传递以下参数: $x + 38, y + 72, 44, 44$ 。
- e) **在 `drawMole` 方法中绘制鼹鼠的眼睛** 在第 33 行, 调用 g 的 `setColor` 方法, 设置鼹鼠眼睛的颜色。将常量 `Color.YELLOW` 传递给 `setColor` 方法。在第 35 行, 加入一行注释, 说明开始绘制鼹鼠的眼睛。随后在第 36 行, 调用 g 的 `fillOval` 方法并为其传递以下参数: $x + 47, y + 84, 8, 8$ 。在第 37 行, 调用 g 的 `fillOval` 方法并为其传递以下参数: $x + 65, y + 84, 8, 8$ 。
- f) **在 `drawMole` 方法中绘制鼹鼠的鼻子** 在第 39 行, 调用 g 的 `setColor` 方法, 设置鼹鼠鼻子的颜色。将常量 `Color.BLACK` 传递给 `setColor` 方法。在第 40 行, 调用 g 的 `fillOval` 方法并为其传递以下参数: $x + 58, y + 97, 5, 5$ 。
- g) **保存应用程序** 保存修改后的源代码文件。

① 需要从 Internet 上下载棒打鼹鼠 (Whack A Mole) 游戏的读者需多加小心, 有一款受病毒感染的版本会在游戏运行过程中读取游戏者硬盘上的数据。

- h) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\WhackAMole` 进入到当前工作目录中。
 - i) 编译应用程序 键入 `javac WhackAMole.java Mole.java` 编译该应用程序。
 - j) 运行完成后的应用程序 若能正确编译应用程序，键入 `java WhackAMole` 来运行它。试着多玩几次游戏以测试该应用程序，确保其结果类似于图 27.37 所示。
 - k) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
 - l) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。
- 27.14 (说出这段代码的作用) 下段代码的运行结果是什么？假定所使用的类均来自绘图应用程序，且此方法位于 `PainterJPanel` 类的内部。

```

1 private void drawJButtonActionPerformed((ActionEvent event)
2 {
3     MyOval oval;
4     for (int i = 0 ; i <= 50 ; i += 10 )
5     {
6         oval = new MyOval( i, 20, 10 , 10 , Color.GREEN );
7         shapes.add( oval );
8     }
9 } // end for
10
11 repaint();
12
13
14 } // end method drawJButtonActionPerformed

```

- 27.15 (找出代码中的错误) 请找出下段代码中的错误。以下代码针对某个 `JButton` 定义了一个 `actionPerformed` 事件处理程序。该事件处理程序能够在 `JPanel` 上绘制一个矩形。假定所使用的类均来自绘图应用程序。

```

1 private void drawImageJButtonActionPerformed( ActionEvent event )
2 {
3     // set shape
4     MyShape rectangle = new MyRectangle( 2 , 3 , 40 , 30 );
5
6     // set color
7     rectangle.setColor( Color.ORANGE );
8
9     // add rectangle to shapesArrayList
10    shapesArrayList.add( rectangle );
11
12 } // end method drawImageJButtonActionPerformed

```

挑战题

- 27.16 (移动图形应用程序) 改进本教程中创建的绘图应用程序，改进后的应用程序在绘制完一个图形以后，该图形便开始以随机速度进行移动，当碰到 `PaintJPanel` 的边界时图形还可反弹回来。应用程序的最终结果可能类似于图 27.38 所示。



图 27.38 移动图形应用程序的 GUI

- a) **将模板复制到工作目录中** 将 C:\Examples\Tutorial27\Exercises\MovingShapes 目录复制到 C:\SimplyJava 目录中。
- b) **打开模板文件** 在自己的文本编辑器中打开 MyMovingShape.java 文件。
- c) **在 MyMovingShape 类中添加改变图形位置的方法** 此继承结构中的抽象超类已被重新命名为 MyMovingShape。为该类添加一个 public 方法 moveShape，该方法无参数及返回值。在 MyMovingShape 类中已为读者新添加了两个实例变量 dx 和 dy。变量 dx 存储图形沿 x 轴方向一次移动的距离，变量 dy 存储图形沿 y 轴方向一次移动的距离。将 dx 加到 x1 和 x2 上，将 dy 加到 y1 和 y2 上。注意使用 get 方法和 set 方法这一好的编程习惯，而不要直接去修改变量的值。
- d) **完成 moveShape 方法** 在 moveShape 方法中添加两条 if 语句，完成图形碰到边界时对其执行的反向处理。第一个 if 语句检查 x 坐标 (x1 或 x2) 是否小于 0 或大于 400。如果为 true，将 dx 的值取负。完成此项操作时，确保使用正确的 get 方法或 set 方法。第二个 if 语句检查 y 坐标 (y1 或 y2) 是否小于 0 或大于 340。如果为 true，将 dy 的值取负。同样，完成此项操作时应确保使用正确的 get 方法或 set 方法。
- e) **保存应用程序** 保存修改后的源代码文件。
- f) **打开模板文件** 在自己的文本编辑器中打开 PaintJPanel.java 文件。
- g) **修改 moveTimerActionPerformed 方法** moveTimerActionPerformed 方法需要对 shapeArrayList 中的每一个图形进行迭代并调用各自图形的 moveShape 方法。完成此项操作，需首先声明一个 MyMovingShape 类型的局部变量 nextShape。然后再声明一个 Iterator 类型的局部变量 shapesIterator，利用调用 shapeArrayList 的 iterator 方法所返回的值对其进行初始化。随后，创建一个 while 循环，判断条件为调用 shapesIterator 的 hasNext 方法所返回的 boolean 值。在 while 循环体内，将调用 shapesIterator 的 next 方法所返回的一个引用赋予 nextShape。next 方法返回 shapeArrayList 中的下一个对象，它可以是 MyLine，MyRectangle 或 MyOval 类型。因此，读者需要将返回的对象构造为 MyMovingShape 对象，然后存储在一个 MyMovingShape 类型的变量中。在结束 while 循环之前，调用 nextShape 的 moveShape 方法。这个 while 循环通过迭代 shapeArrayList 中的每一个图形，调用各自的 moveShape 方法。
- h) **保存应用程序** 保存修改后的源代码文件。
- i) **打开命令提示符窗口改变目录** 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\MovingShapes 进入到当前工作目录中。
- j) **编译应用程序** 通过键入 javac MovingShapes.java PaintJPanel.java MyMovingShape.java 编译该应用程序。
- k) **运行完成后的应用程序** 若能正确编译应用程序，通过键入 java MovingShapes 来运行它。绘制三种不同图形并为每一种图形选择不同的颜色，完成对应用程序的测试。确保能正常移动所有图形，碰到边界时也能反弹回来。
- l) **关闭应用程序** 点击关闭按钮关闭正在运行的应用程序。
- m) **关闭命令提示符窗口** 点击命令提示符窗口的关闭按钮将其关闭。

教程 28 电话号码簿应用程序

Java 语音 API 简介



教学目标

在本教程中，读者将学到以下内容：

- 下载并安装运行 Java 语音应用程序的 FreeTTS 软件
- 通过多媒体技术改进 Java 应用程序
- 在 Java 应用程序中运用 Java 语音 API

计算机刚出现的时候，体积庞大、价格昂贵，人们主要利用它们进行算术计算。使用各式各样媒体——包括图形、动画、视频以及声响效果的多媒体应用程序，却因计算机本身造价高昂、速度缓慢等因素变得难以实现。然而，如今低廉、超高速的处理器已使基于多媒体的应用程序成为了很平常的事物。随着多媒体需求市场的膨胀，用户购买的计算机，其处理器速度却正在变得越来越快，内存容量也是越来越大，而支持多媒体应用程序的带宽则变得越变越宽。

用户都希望看到更多令人兴奋的新型三维多媒体应用程序，并通过动画、音频以及视频等进行交互。设计多媒体程序，同样也是一个令人感到愉悦的创新领域，但与此同时，也会面临更多的挑战。Java 让开发人员将诸如多媒体的能力添加进自己的应用程序之中。

在本教程中，将探索一个可利用文本输入产生合成语音的 Java 语音 API。之后，还将创建一个电话号码簿应用程序，通过 JComboBox 选择一个姓名，应用程序便能够读出其对应的电话号码。

28.1 Java 语音 API

这个电话号码簿应用程序必须满足下面的需求：

应用程序需求分析

某软件公司客户服务代表采用打电话方式同客户进行联络。客户服务代表希望通过一个更为便捷的方法来检索客户的电话号码。为他们开发一个用于存储并检索客户姓名和电话号码的应用程序。客户服务代表同时还希望该应用程序可通过运用语音合成技术（使用 Java 语音 API）帮助他们选择客户的姓名，从而检索出所需要的电话号码。

Java 语音 API 是一项可为应用程序或 Web 页面中添加支持人们同计算机之间实现语音交互能力的技术。Java 语音 API 支持两种类型的核心技术：语音合成和语音识别。语音合成，也称文本到语音的技术，用于从文本中产生合成的语音。语音识别，则是从包含语音的音频输入中产生文本。

关于 Java 语音 API 的基本信息请浏览 Sun 公司的网站：

java.sun.com/products/java-media/speech/

Java 语音 API 可让用户通过说话的方式同应用程序和 Web 页面进行交互。当用户对准麦克风说话时,相应控制会启用语音识别引擎,语音识别引擎是一个能够将输入进麦克风中的声音翻译成计算机可以识别的某种语言的应用程序。Java 语音 API 同时还使用了一个文本到语音的引擎,允许应用程序把输出的文本转化成语音。文本到语音的引擎是一个能够将所键入的单词转换成可通过连接在计算机上的耳机或扬声器进行收听的应用程序。

Sun 并未提供任何有关 Java 语音 API 的实现工具。可以使用其他机构开发的工具,如 Speech Integration Group of Sun Microsystems Laboratories 与来自 IBM 的 Speech for Java 这两个机构合作开发了一个名为 FreeTTS 的软件。并且,Speech for Java 通过结合 IBM 的语音技术,还将语音合成和语音识别技术合并到了用户的接口之中。在本教程中,将使用这个已得到实现的 FreeTTS 软件,它是一个完全由 Java 所编写的开放源代码语音合成软件。开放源代码软件可免费下载并作为完整的应用程序进行使用,或者是作为进一步开发及修改而参考的源代码使用。在读者的计算机中,需要有一个扬声器和一个能够产生识别语音的声卡。

自测题

1. _____ 可将已输入的单词转换成声音。
a) 语音识别引擎 b) 文本到语音的引擎 c) 字符动画集 d) 以上答案都正确
2. 能将麦克风中所输入的语音翻译成计算机可以理解的某种语言的应用程序被称为 _____。
a) 语音识别引擎 b) 文本到语音的引擎 c) 字符动画集 d) 以上答案都正确

答案: 1) b 2) a

28.2 下载并安装 FreeTTS

FreeTTS 软件是 Speech Integration Group of Sun Microsystems Laboratories 针对 Java 语音 API 的一个实现。本教程将演示如何通过使用 FreeTTS 来构建电话号码簿应用程序。为了能运行本教程中的应用程序,必须从发布 FreeTTS 的 Web 网站上下载并安装 FreeTTS。FreeTTS 的下载地址为:

`prdownloads.sourceforge.net/freetts/freetts-1_1_2.tar.gz?download`

登录以后,选择离读者最近的一个下载位置,然后点击最右侧一列中的下载符号对其进行下载(如图 28.1 所示)。将文件 `freetts-1_1_2.tar.gz` 保存在计算机中。`freetts-1_1_2.tar.gz` 文件中含有安装 FreeTTS 所需的一个压缩文档。



图 28.1 定位并下载 FreeTTS

安装 FreeTTS

- 1. 打开 freetts-1_1_2.tar.gz 使用一个能够打开压缩文档的软件，如 WinZip (www.winzip.com) 解开文件 freetts-1_1_2.tar.gz。
- 2. 解压缩 freetts-1_1_2.tar 当如图 28.2 所示的屏幕出现时，通过点击 Yes 按钮，将 freetts-1_1_2.tar 文件解压缩到一个临时目录中。

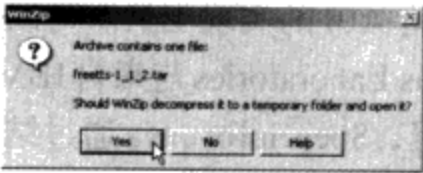


图 28.2 解压 freetts-1_1_2.tar

- 3. 展开 freetts-1_1_2.tar 中的内容 点击图 28.3 中的 Extract 按钮，取出 freetts-1_1_2.tar 中相应的内容。在图 28.4 中，选择 C:\ 目录作为它的目标地址（最终将在目录 C:\ 中创建一个 FreeTTS 目录）。
- 4. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\FreeTTS\lib 进入到 FreeTTS\lib 目录中。
- 5. 安装 Java 语音 API 键入 jsapi.exe，安装 Java 语音 API。接受所出现的许可协议（如图 28.5 所示）。当安装完成以后，会出现一个如图 28.6 中所示的窗口。点击 Close 按钮，将安装窗口关闭。

点击Extract按钮提取
freetts-1_1_2.tar
中的内容

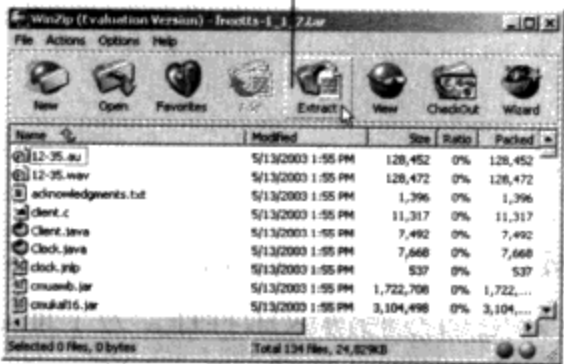


图 28.3 展开 freetts-1_1_2.tar 中的内容

将freetts-1_1_2.tar中
的内容放入目录C:\中

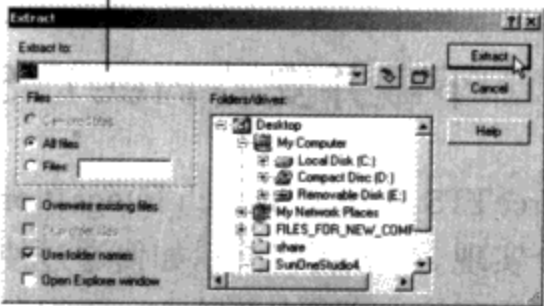


图 28.4 指定目标地址

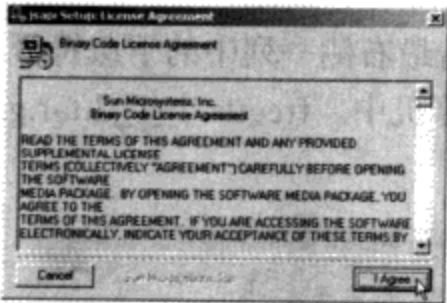


图 28.5 接受许可协议

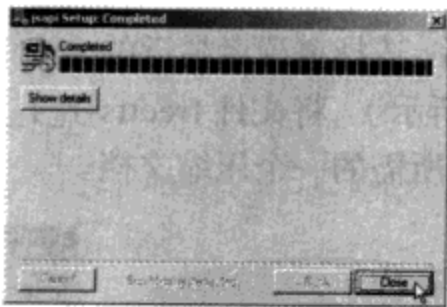


图 28.6 完成 Java 语音 API 的安装

- 6. 安装 speech.properties 文件 将 C:\FreeTTS\speech.properties 文件复制到 C:\j2sdk1.4.1_02\jre\lib 目录中。假如读者安装 J2SE1.4（参考本书前面“准备工作”）的目录并非是 C:\j2sdk1.4.1_02，则应将 speech.properties 文件复制到安装 J2SE 所在目录的 jre\lib 子目录下。

28.3 探试电话号码簿应用程序

前面曾经讲过，即将创建的这个电话号码簿应用程序能够允许用户通过使用 Java 语音 API 中的语音合成技术查找到所需要的电话号码。我们将以这个完成后的应用程序的探试作为起点。之后，学习另外的一些 Java 技术并最终创建出一个属于自己的应用程序。



探试电话号码簿应用程序

- 1. 定位到完成后的应用程序 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\Examples\Tutorial28\CompletedApplication\PhoneBook` 将目录改变到这个完成后的电话号码簿应用程序的目录下面。
- 2. 设置 CLASSPATH 环境变量 回顾前面曾经讲到过，CLASSPATH 环境变量用于指明某应用程序运行时，其所需类库的位置。在命令提示符窗口中通过键入 `SetClasspath.bat`，设置相应的 CLASSPATH 环境变量（注意：如果安装 FreeTTS 的目录并非是目录 `C:\FreeTTS`，则需要在文件 `SetClasspath.bat` 中将 `C:\FreeTTS` 替换成安装 FreeTTS 所在的目录）。这里所提供的 `SetClasspath.bat` 文件是一个包含设置 CLASSPATH 环境变量命令的批处理文件。这样一来，读者只需输入一个批处理文件的名字而无需输入整个命令。
- 3. 运行应用程序 在命令提示符窗口下键入 `java PhoneBook` 来运行这个应用程序（如图 28.7 所示）。从 JComboBox 中选择一个姓名，然后点击 Get Phone Number JButton，这时便能够收听到一个电话号码。



图 28.7 电话号码簿应用程序

- 4. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- 5. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

28.4 创建电话号码簿应用程序

接下来，将通过使用 Java 语音 API 中的语音合成技术，创建该电话号码簿应用程序。通过点击 JComboBox 选取某个姓名，然后收听到指定的电话号码。下列伪代码描述的是，当点击 Get Phone Number JButton 时，电话号码簿应用程序将进行的基本操作。

当点击 Get Phone Number JButton 时
取得所选人姓名
说出所选人的电话号码

现在，既然我们已对电话号码簿应用程序进行了探试并研究了它的伪代码表示，下面，将使用一张 ACE 表，帮助读者最终把这个伪代码转换成 Java 实现。图 28.8 中列出了该应用程序中一些相应的操作、组件以及事件，从而帮助读者最终完成属于自己版本的这一应用程序。

操作	组件	事件
为用户显示用法说明	instruction1JLabel	当应用程序运行时
	instruction2JLabel	
	getPhoneNumberJButton	
取得所选人姓名	nameJComboBox	当用户点击 Get Phone Number JButton 时
说出所选人的电话号码	Synthesizer	

图 28.8 电话号码簿应用程序的 ACE 表

现在，我们已经理解了电话号码簿应用程序的意图，下面就准备开始来创建它。首先，需要把一个合成器添加到应用程序中。

将语音合成器添加至应用程序当中

1. 将模板复制到工作目录中 将 C:\Examples\Tutorial28\TemplateApplication\PhoneBook 目录复制到 C:\SimplyJava 目录中。
2. 打开模板应用程序 在自己的文本编辑器中打开模板文件 PhoneBook.java。
3. 导入 Java 语音 API 包 将图 28.9 中第 7 行至第 8 行添加到程序代码中,从而导入 Java 语音 API 所使用的包 javax.speech 和包 javax.speech.synthesis。javax.speech 包中的类及其接口,将用来支持音频连通性。而 javax.speech.synthesis 包中的类及其接口,则是用来支持语音合成的。通过这些导入的包,开发人员可在执行语音合成及访问包中的类时,省去对 javax.speech 和 javax.speech.synthesis 包名的输入。

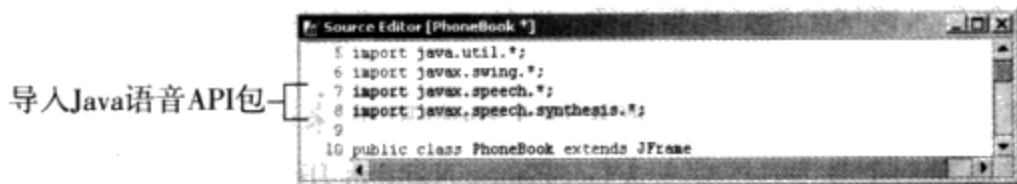


图 28.9 导入 Java 语音 API 包

4. 为语音合成器声明实例变量 将图 28.10 中第 22 行至第 23 行添加到代码中,声明用做读出电话号码的实例变量 speechSynthesizer。读者将在本教程稍后的内容中学习更多有关 Synthesizer 对象的知识。

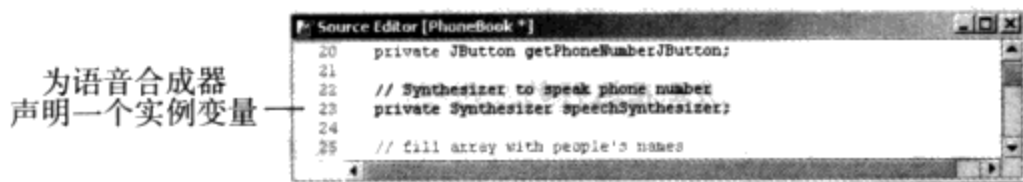


图 28.10 为语音合成器声明一个实例变量

5. 初始化语音合成器 将图 28.11 中第 35 行至第 51 行添加到程序代码中,初始化语音合成器。第 39 行至第 41 行创建了一个 SynthesizerModeDesc 对象,这是一个可用于指明语音合成器属性的描述符。这里所指的属性,包括语音引擎的名称、语音引擎的操作模式、语音引擎所支持的语言、语音引擎的运行状态,以及语音引擎的发音能力,等等。

SynthesizerModeDesc 的构造方法(参见第 39 行至第 41 行)需接收 5 个参数,每一个参数代表语音引擎的一个基本属性。第一个参数(为一个 String)用于指明文本到语音引擎的名称。读者采用的是安装 FreeTTS 时自带的那个引擎,它的名称为 "Unlimited domain FreeTTS Speech Synthesizer from Sun Labs"。第二个参数(为一个 String)用于标识语音引擎的操作模式——若第二个参数被设置为 null,则意味该语音引擎的操作模式不存在特定的要求。第三个参数(为一个 Locale)用于指明该语音引擎所支持的语言。Locale 对象代表世界上的一个特定区域。常量 Locale.US 为代表美国的一个 Locale 对象,因而,与这一 Locale 对象相关联的语言即为英语。第四个参数(为一个 Boolean)用于表示是否有一个正在运行的语音引擎。若传递常量 Boolean.FALSE,则表明不会去选择已得到运行的引擎。第五个参数(为一个 Voice 数组对象,允许开发人员指定某一语音合成器所输出的语音)用于指定语音引擎的声音。若将第五参数设置为 null,则意味该语音引擎的声音并不存在某些特定的要求。

第 44 行通过调用 Central 对象的方法 createSynthesizer,创建出了一个语音合成器。Central 的这个方法 createSynthesizer,需接收一个 SynthesizerModeDesc(描述符)并返回同描述符中所指定属性相匹配的一个 Synthesizer 对象。Synthesizer 对象提供了语音合成的能力,如朗读某一文本。而类 Central 则提供用来访问所有语音输入和输出的能力,如语音合成。同时,类 Central 也可用做定位语音引擎,并根据描述符所定义的属性集合选择出一个相匹配的引擎,从而创建出语音识别器和语音合成器。Central 的 createSynthesizer 方法会在 Central 对象无法创建引擎,或者是 SynthesizerModeDesc 所指定的合成器属性不可知时,抛出相应的异常。第 48 行至第 50 行通过使用 Exception 对象,捕获所有的可能由 createSynthesizer 方法所抛出的异常。

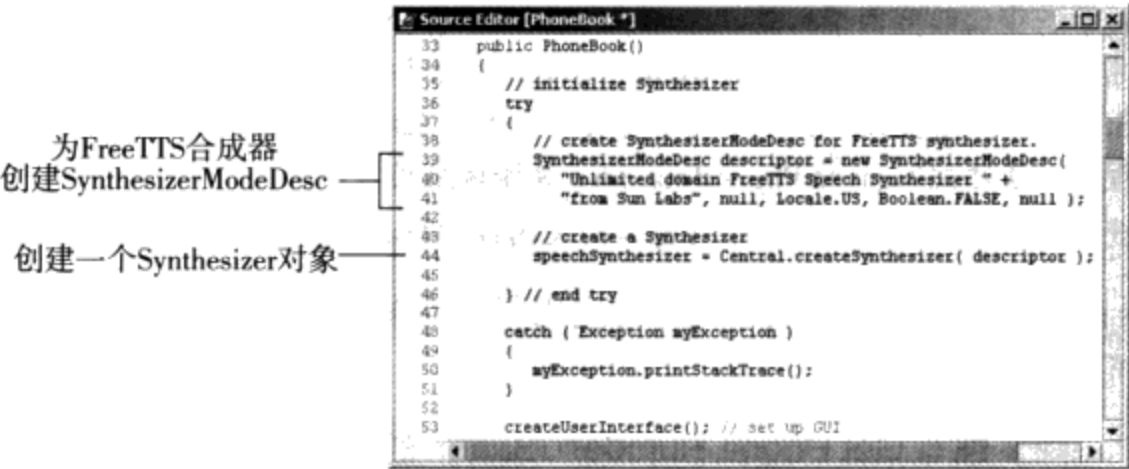


图 28.11 初始化语音合成器

6. 准备用于朗读的 Synthesizer 对象 将图 28.12 中第 46 行至第 59 行添加到 try 语句块中（参见第 36 行至第 61 行）。如果 Central 对象能够成功地创建一个语音合成器，那么，第 47 行至第 53 行中的 Synthesizer 对象的 allocate 方法和 resume 方法将分配该语音引擎所需要的资源并取得可用于朗读的语音合成器。如果 Central 对象没能创建出语音合成器，则第 55 行至第 59 行将显示一个错误的信息并通过调用 System 的 exit 方法终止这一应用程序的执行。

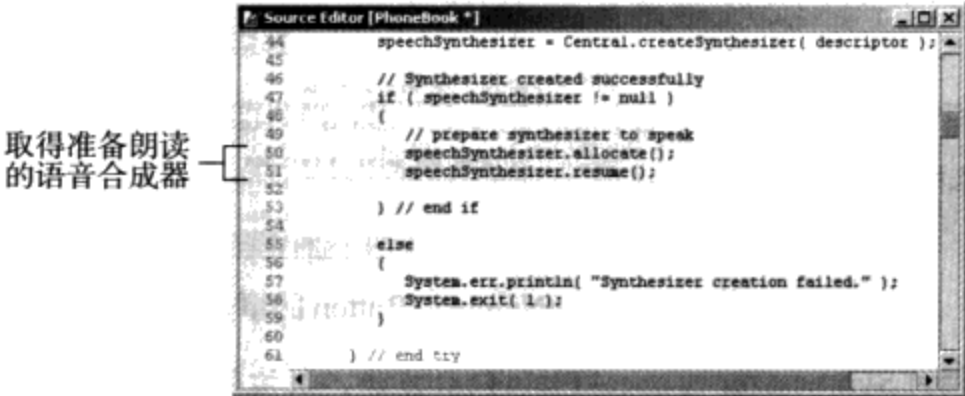


图 28.12 取得准备朗读的 Synthesizer 对象

7. 设置 Synthesizer 对象的属性 将图 28.13 中第 53 行至第 58 行添加到前一步骤（参见第 47 行至第 53 行）中的 if 语句体内。第 54 行至第 55 行通过调用 Synthesizer 对象的 getSynthesizerProperties 方法，获取了同该语音合成器相关联的 SynthesizerProperties 对象。一个 SynthesizerProperties 对象将包含语音合成器的多种属性。每一个属性都可通过调用 setProperty 方法进行改变。比如，通过调用 SynthesizerProperties 的 setSpeakingRate 方法，便可设置该语音合成器的 speakingRate 属性。speakingRate 属性表示每分钟朗读单词的速率。第 58 行通过调用 SynthesizerProperties 对象的 setSpeakingRate 方法，对语音合成器的 speakingRate 属性进行设置。setSpeakingRate 方法接收一个参数——作为指定语音合成器说话速率的 float 型数据。第 58 行将 float 型数据 100.0f 传递给了方法 setSpeakingRate。

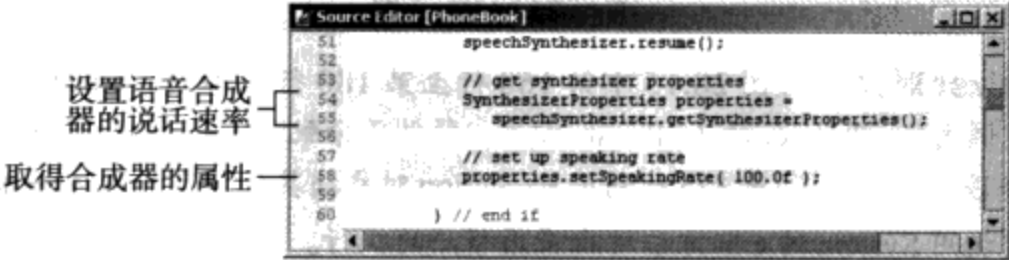


图 28.13 设置 Synthesizer 对象的属性

8. 保存应用程序 保存修改后的源代码文件。

9. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\PhoneBook` 进入到当前的工作目录中。
10. 设置 CLASSPATH 环境变量 在命令提示符窗口中键入 `SetClasspath.bat`, 设置 CLASSPATH 环境变量。
11. 编译应用程序 通过键入 `javac PhoneBook.java` 编译该应用程序。
12. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

既然已将语音合成器添加到了电话号码簿应用程序当中,下面,将添加针对用户点击 Get Phone Number JButton 时所执行的代码。

编写 Get Phone Number JButton 的事件处理程序

1. 编写 getPhoneNumber JButton 的事件处理程序 将图 28.14 中第 158 行至第 166 行添加到方法 `getPhoneNumberJButtonActionPerformed` 中。第 159 行通过调用 `JComboBox` 的 `getSelectedIndex` 方法,取得了某人姓名的一个索引。此索引同该人在 `namesArray` 中的索引,以及在 `numbersArray` 中的电话号码相对应。`namesArray` 和 `numbersArray` 都是程序模块中已得到初始化的实例变量。第 162 行至第 163 行用于从 `namesArray` 以及 `numbersArray` 中取得所选人的姓名和他的电话号码。

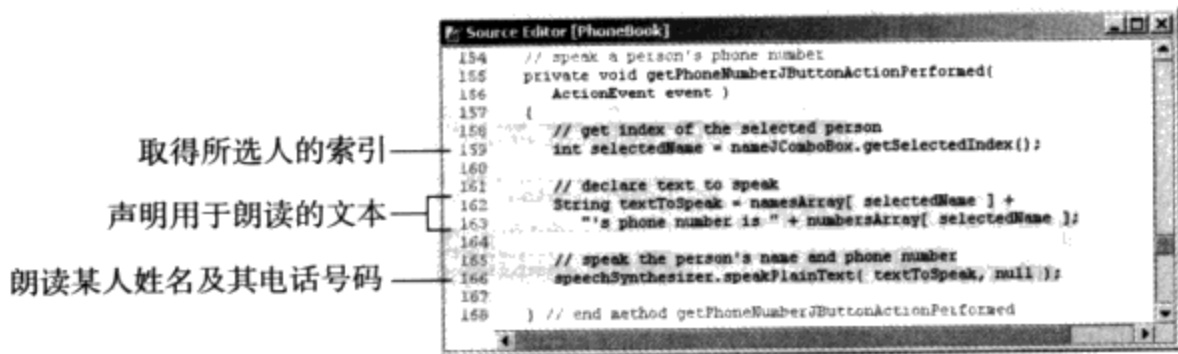


图 28.14 编写 getPhoneNumber JButton 的事件处理程序

第 166 行通过调用 `Synthesizer` 的 `speakPlainText` 方法,朗读第 162 行至第 163 行中所创建的文本。`speakPlainText` 方法将接收两个参数——分别作为指明所朗诵文本的一个 `String`, 以及当合成器朗读时表明不会接收任何语音事件(如到达所朗读文本的末尾)通告的一个 `null`。

2. 保存应用程序 保存修改后的源代码文件。
3. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\PhoneBook` 进入到当前的工作目录中。
4. 设置 CLASSPATH 环境变量 在命令提示符窗口中键入 `SetClasspath.bat`, 设置 CLASSPATH 环境变量。
5. 编译应用程序 通过键入 `javac PhoneBook.java` 编译该应用程序。
6. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

到这里,我们已经添加了 Get Phone Number JButton 的事件处理程序。下面,将编写用户点击应用程序关闭按钮时作为清理语音合成器的代码。

清理合成器对象

1. 清理 Synthesizer 对象 将图 28.15 中第 168 行至第 180 行添加到 `frameWindowClosing` 方法中。第 171 行通过调用 `Synthesizer` 的 `deallocate` 方法,释放已分配给(使用 `allocate` 方法进行调用)语音引擎的资源(如语音合成器)。第 175 行利用 `Exception` 对象捕获任何可能由方法 `deallocate` 抛出的异常。第 179 行通过调用 `System` 的 `exit` 方法终止应用程序的执行。
2. 保存应用程序 保存修改后的源代码文件。
3. 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\PhoneBook` 进入到当前的工作目录中。

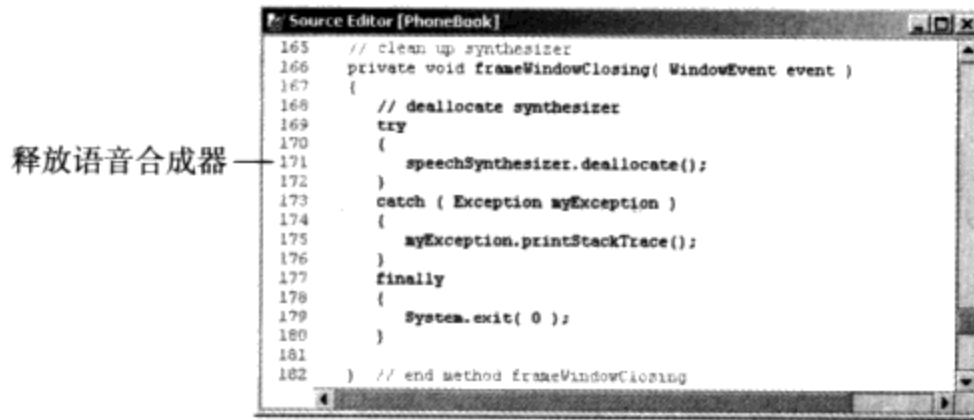


图 28.15 清理合成器

4. 设置 CLASSPATH 环境变量 在命令提示符窗口中输入 SetClasspath.bat, 设置 CLASSPATH 环境变量 (注意: 如果安装 FreeTTS 的目录并非目录 C:\FreeTTS, 则将 SetClasspath.bat 文件中的 C:\FreeTTS 替换成安装 FreeTTS 所在的目录)。
5. 编译应用程序 通过键入 javac PhoneBook.java 编译该应用程序。
6. 运行应用程序 若此应用程序能够正确编译, 通过键入 java PhoneBook 来运行它。图 28.16 中显示了完成后的应用程序的运行结果。

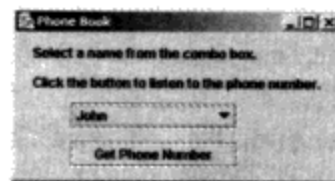


图 28.16 运行完成后的应用程序

7. 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
8. 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

图 28.17 中给出了电话号码簿应用程序的完整源代码。本教程中, 凡需要添加、查看或者是修改的代码, 均在图中相应的代码行中进行了突出显示。

```

1 // Tutorial 28: PhoneBook.java
2 // An application announces phone number with FreeTTS.
3 import java.awt.*;
4 import java.awt.event.*;
5 import java.util.*;
6 import javax.swing.*;
7 import javax.speech.*;
8 import javax.speech.synthesis.*;
9
10 public class PhoneBook extends JFrame
11 {
12     // JLabels to display instructions
13     private JLabel instruction1JLabel;
14     private JLabel instruction2JLabel;
15
16     // JComboBox for names
17     private JComboBox nameJComboBox;
18
19     // JButton to get phone number
20     private JButton getPhoneNumberJButton;
21
22     // Synthesizer to speak phone number
23     private Synthesizer speechSynthesizer;
24

```

导入 Java 语音 API 包

声明实例变量 speechSynthesizer

```

25 // fill array with people's names
26 private String[] namesArray = { "John", "Jennifer", "Howard" };
27
28 // fill array with people's phone numbers
29 private String[] numbersArray =
30     { "(555) 555-9876", "(555) 555-1234", "(555) 555-4567" };
31
32 // no-argument constructor
33 public PhoneBook()
34 {
35     // initialize Synthesizer
36     try
37     {
38         // create SynthesizerModeDesc for FreeTTS synthesizer
39         SynthesizerModeDesc descriptor = new SynthesizerModeDesc(
40             "Unlimited domain FreeTTS Speech Synthesizer " +
41             "from Sun Labs", null, Locale.US, Boolean.FALSE, null );
42         // 为 FreeTTS 合成器创建一个 SynthesizerModeDesc
43         // create a Synthesizer
44         speechSynthesizer = Central.createSynthesizer( descriptor );
45         // 创建一个 Synthesizer 对象
46         // Synthesizer created successfully
47         if ( speechSynthesizer != null )
48         {
49             // prepare synthesizer to speak
50             speechSynthesizer.allocate();
51             speechSynthesizer.resume();
52             // 准备一个作为朗读的合成器
53             // get synthesizer properties
54             SynthesizerProperties properties =
55                 speechSynthesizer.getSynthesizerProperties();
56             // 取得合成器的属性
57             // set up speaking rate
58             properties.setSpeakingRate( 100.0f );
59             // 设置语音合成器的朗读语速
60         } // end if
61     }
62     else
63     {
64         System.err.println( "Synthesizer creation failed." );
65         System.exit( 1 );
66     }
67 } // end try
68 } // end constructor
69
70 catch ( Exception myException )
71 {
72     myException.printStackTrace();
73 }
74
75 createUserInterface(); // set up GUI
76
77 } // end constructor
78
79 // create and position GUI components; register event handler
80 private void createUserInterface()
81 {
82     // get content pane for attaching GUI components
83     Container contentPane = getContentPane();

```

```
84
85 // enable explicit positioning of GUI components
86 contentPane.setLayout( null );
87
88 // set up instruction1JLabel
89 instruction1JLabel = new JLabel();
90 instruction1JLabel.setBounds( 16 , 8 , 264 , 23 );
91 instruction1JLabel.setText(
92     "Select a name from the combo box." );
93 contentPane.add( instruction1JLabel );
94
95 // set up instruction2JLabel
96 instruction2JLabel = new JLabel();
97 instruction2JLabel.setBounds( 16 , 35 , 264 , 23 );
98 instruction2JLabel.setText(
99     "Click the button to listen to the phone number." );
100 contentPane.add( instruction2JLabel );
101
102 // set up nameJComboBox
103 nameJComboBox = new JComboBox( namesArray );
104 nameJComboBox.setBounds( 50 , 65 , 150 , 23 );
105 contentPane.add( nameJComboBox );
106
107 // set up getPhoneNumberJButton
108 getPhoneNumberJButton = new JButton();
109 getPhoneNumberJButton.setBounds( 50 , 100, 150 , 23 );
110 getPhoneNumberJButton.setText( "Get Phone Number" );
111 contentPane.add( getPhoneNumberJButton );
112 getPhoneNumberJButton.addActionListener(
113
114     new ActionListener() // anonymous inner class
115     {
116         // event handler called when getPhoneNumberJButton
117         // is clicked
118         public void actionPerformed((ActionEvent event) )
119         {
120             getPhoneNumberJButtonActionPerformed( event );
121         }
122     } // end anonymous inner class
123 ); // end call to addActionListener
124
125 // set properties of application's window
126 setTitle( "Phone Book" ); // set title bar string
127 setSize( 300, 160 ); // set window size
128 setVisible( true ); // display window
129
130 // ensure synthesizer is cleaned up
131 // when user closes application
132 addWindowListener(
133
134     new WindowAdapter() // anonymous inner class
135     {
136         public void windowClosing( WindowEvent event )
137         {
138             frameWindowClosing( event );
139         }
140     } // end anonymous inner class
141 );
```



```

145     ); // end addWindowListener
146
147 } // end method createUserInterface
148
149 // speak a person's phone number
150 private void getPhoneNumberJButtonActionPerformed(
151     ActionEvent event )
152 {
153     // get index of the selected person
154     int selectedName = nameJComboBox.getSelectedIndex();    获取所选人的索引
155
156     // declare text to speak
157     String phoneNumberString = namesArray[ selectedName ] +    声明需朗
158     "'s phone number is " + numbersArray[ selectedName ];    读的文本
159
160     // speak the person's name and phone number
161     speechSynthesizer.speakPlainText( phoneNumberString, null );    朗读某人姓名及其电话号码
162
163 } // end method getPhoneNumberJButtonActionPerformed
164
165 // cleanup synthesizer
166 private void frameWindowClosing( WindowEvent event )
167 {
168     // deallocate synthesizer
169     try
170     {
171         speechSynthesizer.deallocate();    释放合成器
172     }
173     catch ( Exception myException )
174     {
175         myException.printStackTrace();
176     }
177     finally
178     {
179         System.exit( 0 );
180     }
181
182 } // end method frameWindowClosing
183
184 // main method
185 public static void main( String[] args )
186 {
187     PhoneBook application = new PhoneBook();
188
189 } // end method main
190
191 } // end class PhoneBook

```

图 28.17 电话号码簿应用程序

自测题

- 当语音引擎其朗诵语速过快时，可通过调用 _____ 方法使语音引擎的语速放慢。
a) setSpeakingRate b) changeSpeakingRate c) speakSlowly d) 以上答案都不对
- Synthesizer 的 _____ 方法用做朗读文本。
a) speakString b) speakText c) speakPlainText d) 以上答案都不对

答案：1) a 2) c

28.5 小结

本教程介绍了有关 Java 语音 API 的知识，使读者懂得如何通过 Java 语音 API 改进软件应用程序。同时，还了解了有关 FreeTTS 以及如何对其进行安装等细节。之后，学习了如何编写使用 Java 语音 API 的相关代码。

通过使用 Java 语音 API，可利用语音合成技术中的朗读方式，创建一个能够同用户进行交互的应用程序。主要步骤是先使用方法 `createSynthesizer` 创建一个 `Synthesizer` 对象。与此同时，还可利用 `setSpeakingRate` 方法改变语音合成器的属性。接着，通过 `Synthesizer` 对象的 `speakPlainText` 方法可为用户朗读一个电话号码。

在教程 29 至教程 32 中，将提供一个基于 Web 的书店应用程序的案例研究。我们将学习如何构建一个通过使用 Web 浏览器进行访问的应用程序。同时，还将了解到有关应用程序的三层概念，即应用程序被分为三个主要独立部分，这些部分可驻留在同一台计算机上或者是通过一个网络（如 Internet），分布在多个不同的计算机上。

技术小结

安装 FreeTTS

- 打开 `freetts-1_1_2.tar.gz` 文件。
- 解压缩 `freetts-1_1_2.tar` 文件。
- 展开 `freetts-1_1_2.tar` 文件的内容。
- 安装 Java 语音 API。
- 安装 `speech.properties` 文件。

创建一个 Synthesizer 对象

- 创建一个 `SynthesizerModeDesc` 对象，该对象将作为指明语音合成器基本属性的一个描述符。
- 通过 `SynthesizerModeDesc` 对象，调用 `Central` 对象的 `createSynthesizer` 方法。

准备一个用于朗读的 Synthesizer 对象

- 调用 `Synthesizer` 对象的 `allocate` 方法。
- 调用 `Synthesizer` 对象的 `resume` 方法。

获取 Synthesizer 对象的属性

- 通过调用 `Synthesizer` 对象的 `getSynthesizerProperties` 方法，取得包含语音合成器不同属性的一个 `SynthesizerProperties` 对象。

设置语音合成器的 speakingRate 属性

- 调用 `SynthesizerProperties` 对象的 `setSpeakingRate` 方法，其中，表示每分钟单词朗读速率的 `float` 值可用来设置语音合成器的 `speakingRate` 属性。

使语音合成器朗读语音

- 通过调用 `Synthesizer` 对象的 `speakPlainText` 方法，朗读纯文本语音。

释放已分配给 Synthesizer 对象的资源

- 调用 `Synthesizer` 对象的 `deallocate` 方法，释放已分配给语音引擎的资源（如语音合成器）。

关键术语

Synthesizer 的 allocate 方法 分配语音引擎所需的资源。

Boolean.FALSE 可将 Boolean 类中的一个 false 值表示成 Boolean 对象的一个常量。

Central 类 可提供访问针对语音输入和输出的所有能力, 比如说语音合成。还可做: 定位语音引擎; 根据描述符定义的属性集合选择一个相匹配的引擎; 创建语音识别器和语音合成器。

Central 的 createSynthesizer 方法 用来创建一个语音合成器。

Synthesizer 的 deallocate 方法 用做释放已分配给语音引擎的资源 (如语音合成器)。

Synthesizer 的 getSynthesizerProperties 方法 用来取得同语音合成器相关联的 SynthesizerProperties 对象。

javax.speech 包 javax.speech 包中包含用于支持音频连通性的所有类及其接口。

javax.speech.synthesis 包 一个由支持语音合成技术的类及其接口组合而成的包。

Java 语音 API 用做向应用程序或 Web 页面中添加语音功能, 从而支持人与计算机之间实现交互的一种新型方式。

Java 语音技术 通过语音合成技术从文本中产生合成语音的一项技术。

Locale 对象 代表世界上某个特定区域的一个对象。

Locale.US 常量 代表美国的一个 Locale 对象。

多媒体 利用各种媒体, 如声音、视频以及动画, 产生应用程序当中的相应内容。

开放源代码软件 可以免费下载的软件, 可作为完整的应用程序来使用, 也可作为创建及修改的源代码来使用。

Synthesizer 的 resume 方法 使语音合成器准备进行朗读的一个方法。

SynthesizerProperties 的 setSpeakingRate 方法 用来设置语音合成器的 speakingRate 属性。

语音合成器的 speakingRate 属性 用来指明每分钟所念单词速率的一个属性。

Synthesizer 的 speakPlainText 方法 为用户朗读出文本的方法。

语音识别 一项能够从包含语音的音频输入中产生文本的技术。

语音识别引擎 一种能够将麦克风中所输入的口音翻译成计算机所能理解的语言的应用程序。

语音合成 也称文本到语音的技术, 即从文本中产生合成的语音。

Synthesizer 对象 一种可提供语音合成能力 (如朗读文本) 的对象。

SynthesizerModeDesc 对象 指示语音合成器基本属性的描述符对象。

SynthesizerProperties 对象 包含语音合成器各种属性的对象。

语音到文本的引擎 将所输入单词翻译成用户可通过耳机或者连接在计算机上的扬声器来收听语音的一个应用程序。

文本到语音的技术 通过文本产生合成语音的一项技术。

Voice 类 指明某合成器所输出语音的一个类。

Java 类库索引

Central 此类 (位于 javax.speech 包内) 提供了所有针对语音输入和输出 (如语音合成器) 的访问。

● 方法

createSynthesizer 创建一个 Synthesizer 对象, 该对象作为向用户朗读文本时使用。

Locale 此类代表世界上某个特定的区域。

● 常量

Locale.US 代表美国的一个 Locale 对象。

Synthesizer 此类可提供语音合成能力, 如朗读文本。

● 方法

allocate 分配语音引擎所需的资源, 比如, 语音合成器。

deallocate 释放已分配给语音引擎的资源, 比如, 语音合成器。

getSynthesizerProperties 取得同语音引擎相关的 SynthesizerProperties 对象。

resume 用于取得准备朗读的语音引擎。

speakPlainText 向用户朗读文本。

SynthesizerModeDesc 此类具体指定了语音合成器的基本属性。

● 构造方法

SynthesizerModeDesc 接收 5 个参数，每一个参数代表语音引擎的一个基本属性。第 1 个参数（为一个 **String**）指明文本到语音引擎的名称。第 2 参数（为一个 **String**）用于标识语音引擎的操作模式。第 3 个参数（为一个 **Locale**）用于指明该语音引擎所支持的语言。第 4 个参数（为一个 **Boolean**）表示是否有一个正在运行的语音引擎。第 5 个参数（为一个 **Voice** 数组对象）代表语音引擎的声音。

```
SynthesizerModeDesc descriptor = new SynthesizerModeDesc(
    "Unlimited domain FreeTTS Speech Synthesizer " +
    "from Sun Labs", null, Locale.US, Boolean.FALSE, null );
```

SynthesizerProperties 此类包含语音合成器的各种属性。

● 属性

speakingRate 指定每分钟朗读单词的速率。

● 方法

setSpeakingRate 设置语音合成器的 **speakingRate** 属性。

Voice 使用此类指定语音合成器的声音输出。

习题

选择题

- 28.1 _____ 方法用来指定语音合成器所朗读的内容。
a) **speakPlainText** b) **say** c) **voice** d) 以上答案均不对
- 28.2 _____ 方法用来取得语音引擎所需的资源。
a) **getResource** b) **obtainResource** c) **allocate** d) 以上答案均不对
- 28.3 方法 **createSynthesizer** 需接收一个表示 _____ 的参数。
a) 语音合成器位置 b) 语音合成器名称
c) 语音合成器声音 d) 指明语音合成器属性描述符
- 28.4 **Synthesizer** 的 _____ 方法可返回代表语音合成器属性的一个对象。
a) **getSynthesizerProperty** b) **getSynthesizerProperties**
c) **getProperty** d) **getProperties**
- 28.5 利用 _____ 方法释放分配给语音引擎的资源。
a) **deallocate** b) **freeResource** c) **releaseResource** d) 以上答案均不对
- 28.6 **speakingRate** 属性根据 _____ 指定朗读速率。
a) 每秒钟的字符数 b) 每分钟的字符数 c) 每秒钟的单词数 d) 每分钟的单词数
- 28.7 _____ 的对象代表某语音合成器的属性。
a) **SynthesizerProperties** b) **Central** c) **SynthesizerModeDesc** d) 以上答案均不对
- 28.8 _____ 的对象代表世界上某个特定的区域。
a) **Locale.WORLD** b) **Locale** c) **Region** d) 以上答案均不对
- 28.9 通过使用 _____ 包中的类及其接口支持语音合成。
a) **javax.speech** b) **javax.speech.synthesis** c) **java.speech** d) **java.speech.synthesis**
- 28.10 方法 _____ 将返回 **JComboBox** 中用户所需项目的一个索引。
a) **getSelectedItem** b) **getSelected** c) **getSelectedIndex** d) 以上答案均不对

练习题

- 28.11 （使用 Java 语音 API 的预约工作簿应用程序）编写一个应用程序，用户可通过使用 Java 语音 API 向预约工作簿中添加预约（如图 28.18 所示）。当点击 **Add JButton** 时，该应用程序会把约会人姓

名、预约时间及日期分别添加至三个不同的 ArrayList 中。当用户点击 Get Appointments JButton 时, 应用程序便能够读出与在 Appointment With: JTextField 中所显示约会人姓名相匹配的约会时间及日期。

- a) **将模板复制到工作目录中** 将 C:\Examples\Tutorial28\Exercises\AppointmentBook 目录复制到 C:\SimplyJava 目录中。

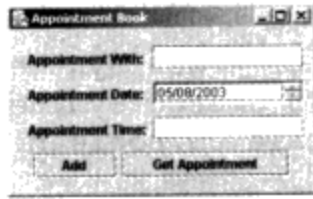


图 28.18 预约工作簿 GUI

- b) **打开模板文件** 在自己的文本编辑器中打开 AppointmentBook.java 文件。
- c) **导入 Java 语音 API 包** 导入 javax.speech 和 javax.speech.synthesis 包。
- d) **声明实例变量** 如第 48 行所示, 声明三个 ArrayList 类型的实例变量, 存储预约的日期、时间及约会人。声明一个类型为 Synthesizer 的实例变量, 作为将来朗读文本时使用。
- e) **创建一个 Synthesizer 对象** 在 AppointmentBook 的构造方法内, 创建一个 Synthesizer 对象。之后, 分配所需的资源并取得准备朗读的合成器。
- f) **向 addJButtonActionPerformed 方法中添加代码** 找到 addJButtonActionPerformed 方法, 该方法紧接 createUserInterface 方法之后。向 addJButtonActionPerformed 方法中添加代码, 将用户所提供的信息添加到与其对应的 ArrayList 之中。而在 Appointment With: JTextField 中的输入, 则需添加至包含与用户进行预约的约会人姓名 ArrayList 中。同时, 预约日期和时间的输入也应添加至各自的 ArrayList 中。如果用户在 Appointment With: 或 Appointment Time: JTextField 中保留空白的话, 则会显示出一条表示错误的信息。
- g) **向 getAppointmentJButtonActionPerformed 方法中添加代码** 找到 getAppointmentJButtonActionPerformed 方法, 该方法紧接 addJButtonActionPerformed 方法之后。向 getAppointmentJButtonActionPerformed 方法中添加代码, 使应用程序能够朗读出与用户在 appointmentWithJTextField 中所列约会人进行预约的时间和日期。如果用户在 Appointment With: JTextField 中保留空白的话, 那么会显示出一条表示错误的信息。假若未安排任何预约, 则应用程序还需要通知用户并无任何安排的预约 (注意: 在 dateJSpinner 中存储的日期应与 "Fri Apr 04 16:22:09 EST 2003" 中所表示的格式相同。对于 FreeTTS 来说, 要朗读出这样的文本是很困难的。因此, 还需调用模板中预先定义好的方法 parseDate 将日期转换成 FreeTTS 可以朗读的文本。例如, 方法 parseDate 可将字符串 "Fri Apr 04 16:22:09 EST 2003" 转换成 "Friday April 04 2003")。
- h) **释放已分配给语音合成器的资源** 找到 frameWindowClosing 方法, 该方法紧接 parseMonth 方法之后。在 frameWindowClosing 方法中, 添加用以释放已分配给语音合成器资源的代码。
- i) **保存应用程序** 保存修改后的源代码文件。
- j) **打开命令提示符窗口改变目录** 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\AppointmentBook 进入到当前的工作目录中。
- k) **设置 CLASSPATH 环境变量** 在命令提示符窗口中键入 SetClasspath.bat 设置 CLASSPATH 环境变量 (注意: 如果安装 FreeTTS 的目录并非目录 C:\FreeTTS, 则需要将 SetClasspath.bat 文件中的 C:\FreeTTS 替换为安装 FreeTTS 所在的目录)。
- l) **编译应用程序** 通过键入 javac AppointmentBook.java 编译该应用程序。
- m) **运行完成后的应用程序** 若能正确编译应用程序, 通过键入 java AppointmentBook 来运行它。在该工作簿中添加一个预约, 然后点击 Get Appointment JButton, 使应用程序能够将相应的信息返回给用户。
- n) **关闭应用程序** 点击关闭按钮关闭正在运行的应用程序。

o) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

28.12 (使用 Java 语音 API 的双骰子游戏应用程序)。修改教程 15 中的双骰子游戏应用程序, 为它添加 Java 语音 API 功能。完成后的应用程序如图 28.19 所示。

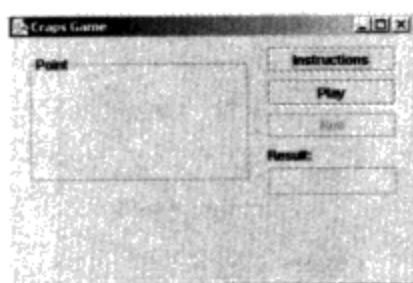


图 28.19 修改后的双骰子游戏应用程序的 GUI

- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial28\Exercises\CrapsGameEnhancement 目录复制到 C:\SimplyJava 目录中。
- b) 打开模板文件 在自己的文本编辑器中打开 CrapsGame.java 文件。
- c) 导入 Java 语音 API 包 导入 javax.speech 和 javax.speech.synthesis 包。
- d) 声明实例变量 在位于 CrapsGame 构造方法的上面, 声明一个类型为 Synthesizer 的实例变量, 作为将来朗读文本时使用。
- e) 创建一个 Synthesizer 对象 在 CrapsGame 构造方法的内部, 创建一个 Synthesizer 对象。之后, 分配相应的资源并准备朗读的合成器。
- f) 向 instructionsJButtonActionPerformed 方法中添加代码 找到 instructionsJButtonActionPerformed 方法, 该方法紧接 createUserInterface 方法之后。向 instructionsJButtonActionPerformed 方法中添加代码, 使语音合成器能够说出此游戏的游戏规则。
- g) 向 playJButtonActionPerformed 方法中添加代码 找到 playJButtonActionPerformed 方法, 该方法紧接 instructionsJButtonActionPerformed 方法之后。向 playJButtonActionPerformed 方法中添加相应的代码。当用户赢得游戏时, 该应用程序将宣布 "Congratulations, you won!"。如果用户失败, 应用程序则宣布 "Sorry, you lost!"。如果用户既没有赢也没有输, 则应用程序宣布 "Please roll again"。
- h) 向 rollJButtonActionPerformed 方法中添加代码 找到 rollJButtonActionPerformed 方法, 该方法紧接 playJButtonActionPerformed 方法之后。在 rollJButtonActionPerformed 方法中添加相应的代码。若用户“抛得一个好点”, 则应用程序应宣布 "Congratulations, you won!"。如果用户滚动的点数为 7, 应用程序将宣布 "Sorry, you lost!"。否则, 应用程序宣布 "Please roll again"。
- i) 释放已分配给语音合成器的资源 找到 frameWindowClosing 方法, 该方法紧接 displayDie 方法之后。在 frameWindowClosing 方法的内部, 添加释放已分配给语音合成器资源的代码。
- j) 保存应用程序 保存修改后的源代码文件。
- k) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\CrapsGameEnhancement 进入到当前的工作目录中。
- l) 设置 CLASSPATH 环境变量 在命令提示符窗口中键入 SetClasspath.bat 设置 CLASSPATH 环境变量 (注意: 如果安装 FreeTTS 的目录并非目录 C:\FreeTTS, 则需将 SetClasspath.bat 文件中的 C:\FreeTTS 替换为安装 FreeTTS 所在的目录)。
- m) 编译应用程序 通过键入 javac CrapsGame.java 编译该应用程序。
- n) 运行完成后的应用程序 若能正确编译应用程序, 键入 java CrapsGame 来运行它。通过点击 Play JButton, 启动双骰子游戏。多次滚动骰子, 查看应用程序是否能够的确说出正确的信息。
- o) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
- p) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。

28.13 (使用Java语音API的安检面板应用程序)修改教程11中的安检面板应用程序,为其添加Java语音API功能。完成后的应用程序如图28.20所示。

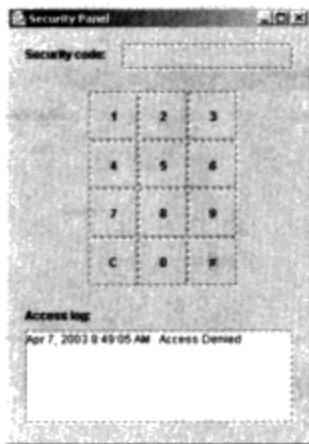


图 28.20 修改后的安检面板应用程序

- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial28\Exercises\SecurityPanelEnhancement 目录复制到 C:\SimplyJava 目录中。
 - b) 打开模板文件 在自己的文本编辑器中打开 SecurityPanel.java 文件。
 - c) 导入 Java 语音 API 包 导入 javax.speech 和 javax.speech.synthesis 包。
 - d) 声明实例变量 在位于 SecurityPanel 构造方法的上面,声明一个类型为 Synthesizer 的实例变量,作为朗读文本时使用。
 - e) 创建一个 Synthesizer 对象 在 SecurityPanel 构造方法的内部,创建一个 Synthesizer 对象。之后,分配相应的资源并准备朗读的合成器。
 - f) 向 enterJButtonActionPerformed 方法中添加代码 找到 enterJButtonActionPerformed 方法,该方法紧接 nineJButtonActionPerformed 方法之后。向 enterJButtonActionPerformed 方法中添加使用该语音合成器的代码。如果用户输入的是一个有效访问码,则应用程序将显示出对用户的欢迎,并读出相应访问码所代表的雇员种类。如果访问码无效,则应用程序会说出这是一个无效的代码,并拒绝任何人进入。
 - g) 释放已分配给语音合成器的资源 找到 frameWindowClosing 方法,该方法紧接 enterJButtonActionPerformed 方法之后。在 frameWindowClosing 方法的内部,添加释放已分配给语音合成器资源的代码。
 - h) 保存应用程序 保存修改后的源代码文件。
 - i) 打开命令提示符窗口改变目录 选择 Start → Programs → Accessories → Command Prompt 打开一个命令提示符窗口。键入 cd C:\SimplyJava\SecurityPanelEnhancement 进入到当前的工作目录中。
 - j) 设置 CLASSPATH 环境变量 在命令提示符窗口中键入 SetClasspath.bat 设置 CLASSPATH 环境变量(注意:如果安装 FreeTTS 的目录并非目录 C:\FreeTTS,则需将 SetClasspath.bat 文件中的 C:\FreeTTS 替换为安装 FreeTTS 所在的目录)。
 - k) 编译应用程序 通过键入 javac SecurityPanel.java 编译该应用程序。
 - l) 运行完成后的应用程序 若能正确编译应用程序,通过键入 java SecurityPanel 来运行它。键入一个正确的访问码,然后按下 # JButton,确认应用程序能够说出正确的信息。完后,再试着输入一个不正确的访问码。这时,应能听到该应用程序会朗读出一条表示未输入正确代码的错误信息。
 - m) 关闭应用程序 点击关闭按钮关闭正在运行的应用程序。
 - n) 关闭命令提示符窗口 点击命令提示符窗口的关闭按钮将其关闭。
- 28.14 (说出这段代码的作用)在点击 Speak JButton 后,下面这个方法将完成怎样的操作? speech-Synthesizer 变量是 Synthesizer 对象的一个引用,并已作为实例变量得到声明。

- h) **释放已分配给语音合成器的资源** 找到 `frameWindowClosing` 方法, 该方法紧接 `calculateMonthlyPayment` 方法之后。在 `frameWindowClosing` 方法的内部, 添加可释放已分配给语音合成器资源的代码。
- i) **保存应用程序** 保存修改后的源代码文件。
- j) **打开命令提示符窗口改变目录** 选择 `Start` → `Programs` → `Accessories` → `Command Prompt` 打开一个命令提示符窗口。键入 `cd C:\SimplyJava\CarPaymentCalculatorEnhancement` 进入到当前工作目录。
- k) **设置CLASSPATH环境变量** 在命令提示符窗口中键入 `SetClasspath.bat` 设置 `CLASSPATH` 环境变量 (注意: 如果安装 `FreeTTS` 的目录并非目录 `C:\FreeTTS`, 则需将 `SetClasspath.bat` 文件中的 `C:\FreeTTS` 替换为安装 `FreeTTS` 所在的目录)。
- l) **编译应用程序** 通过键入 `javac CarPayment.java` 编译该应用程序。
- m) **运行完成后的应用程序** 若能正确编译应用程序, 通过键入 `java CarPayment` 来运行它。
- n) **关闭应用程序** 点击关闭按钮关闭正在运行的应用程序。
- o) **关闭命令提示符窗口** 点击命令提示符窗口的关闭按钮将其关闭。



教程 29 Web 书店应用程序

Web 应用程序的开发及 Apache Tomcat Web 服务器简介

教学目标

在本教程中，读者将学到以下内容：

- 利用 Apache Tomcat Web 服务器向客户提供 Web 服务
- 通过 Web 服务器请求相关文档
- 利用 JSP (Java Server Pages) 技术执行 Web 应用程序

在前面的教程中，我们使用 Java 开发了一些桌面应用程序。对于每一个应用程序，都包含了一个与用户进行交互的 GUI。事实上，我们还可利用 Java 开发 Web 应用程序。这些应用程序（也称基于 Web 的应用程序），使用 JSP (JavaServer Pages) 技术创建 Web 内容——数据可通过 Internet Explorer（来自 Microsoft 公司）或者是 Netscape（来自 Netscape 公司）Web 浏览器来查看。Web 内容包括 HTML (HyperText Markup Language, 超文本标记语言) 文档及相关的图像。JSP 技术就是利用 Java 和 HTML 来开发动态 Web 页面的。其中，HTML 是一门可完全由 Web 浏览器解释并利用 GUI 元素、文本和图像进行在线发布的语言。通过使用 HTML，开发人员可借助超文本链接方式创建出访问其他在线文档的 Web 页面，而且能够设计出一些获取用户输入的表单（类似 JComboBox, JTextField 和 JButton 的组件）。我们将在教程 30 中学习用以创建 Web 书店应用程序的 HTML 语言。

在本教程中，将学习有关创建 Web 应用程序三层架构的内容和一些 Web 服务器的知识，并安装一个 Apache Tomcat Web 服务器，该服务器是运行 Web 书店应用程序所必需的。之后，通过 Web 书店应用程序的运行过程，学习 Web 开发方面的重要概念。这个应用程序由两个 Web 页面组成。第一个页面将显示一系列书目。在选择其中的一本书之后，通过点击页面上的一个按钮可将该浏览器中的当前页面定位到另一个 Web 页面上，所选图书的相关信息也会立即从数据库中检索出来并显示在第二个页面上。该页面中还包含一个链接，点击它，会让 Web 浏览器再次返回到第一个 Web 页面上，从而使用户可重新选择另外一本不同的图书。在创建这个应用程序之前，将会学到一些有关 Web 开发方面的基本概念，这将有助于理解即将探试的 Web 书店应用程序。在教程 30 至教程 32 中，将对这个 Web 书店应用程序的伪代码及其 ACE 表进行分析。之后，通过使用本书中即将学到的一些高级 Java 技术，最终实现这个应用程序。

29.1 多层架构

Web 应用程序属于多层应用程序，有时也被称为 n 层应用程序。多层应用程序可将其自身功能分散在多个相互独立的层（即功能的逻辑分组）当中。应用程序中的这些层，可以位于同一台计算机当中，或者是通过网络（最常见的就是 Internet），分布在多台计算机上。在下面这个案例研究中，

我们将在同一台计算机上实现所有的三层关系。图 29.1 中列出了一个多层应用程序的基本结构。在随后的内容中，将解释每一层所代表的含义。

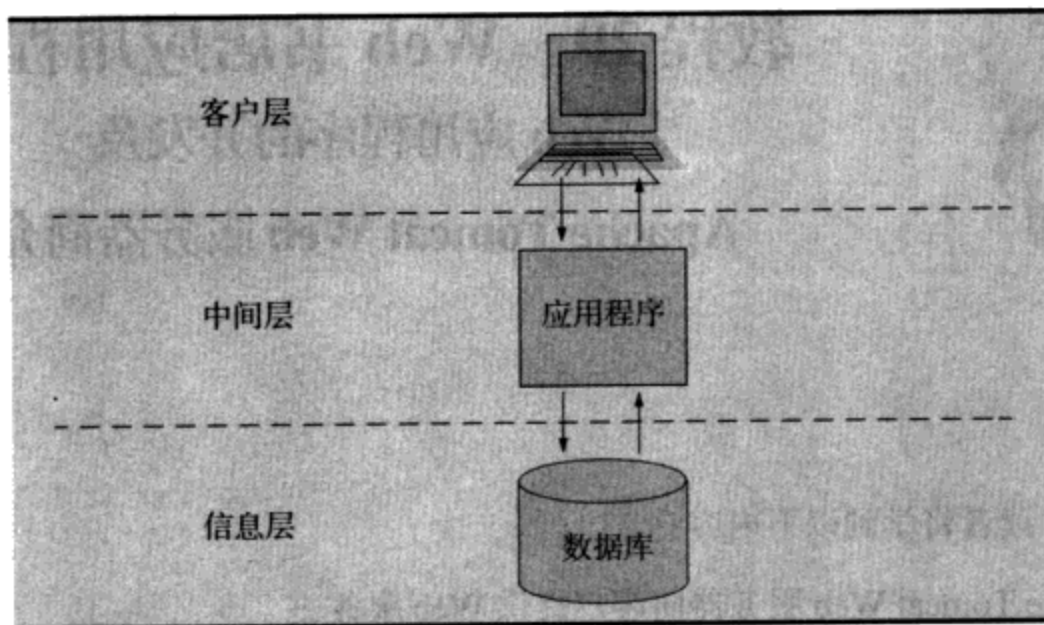


图 29.1 三层应用程序模型

信息层（也叫数据层或底层）作为保留应用程序中的数据。如 Web 书店应用程序的信息层，就是利用包含其相关产品信息的一个数据库来实现的，这些信息包括书名、作者名、出版日期、版本号、ISBN 编码、图书简介以及图书价格等。

中间层用做控制应用程序中客户（如 Web 浏览器）与信息层中的应用程序数据之间的交互。在 Web 书店应用程序中，中间层代码可用来确定用户到底选择的是哪一本图书，以及如何从数据库中检索出该书的相关信息。中间层还可确定所选图书数据的显示格式。所以说，中间层代表了 Web 应用程序的功能实现，因而也常被称为业务逻辑层。

客户层或称顶层，是应用程序的用户接口，通常是一个 Web 浏览器。用户通过客户层（浏览器）输入文本、从列表中选择项目、点击按钮等操作同应用程序进行直接交互。浏览器可以把用户的操作以及用户输入的数据报告给中间层，并利用中间层来处理信息。同时，中间层还可向信息层发出请求并检索出需要的数据。客户层中的 Web 浏览器，随后通过中间层将信息层中所检索出的数据显示给用户。

自测题

1. 在一个典型三层应用程序中，信息层也称 _____ 层。
a) 顶 b) 中间 c) 底 d) 客户
2. 在一个典型三层应用程序中，中间层所充当的角色是 _____。
a) 控制客户层和信息层之间的交互 b) 控制客户和用户接口之间的交互
c) 显示应用程序的用户接口 d) 提供应用程序所需的数据库

答案：1) c 2) a

29.2 Web 服务器

Web 服务器是一种根据请求资源（如 HTML 文档以及相关数据库的访问）响应客户（通过 Web 浏览器）的专用软件。为了从 Web 服务器中获取文档，用户必须知道该文档的地址。URL（Uniform Resource Locator，统一资源定位器）被认为是借助 Web 浏览器定位至某个 Web 资源的一个地址。一个 URL 包括：该资源所使用的协议（如 http）、该资源所在机器的机器名或 IP 地址（本教程将

在随后的内容中讨论有关机器名及 IP 地址的概念), 以及该资源的名称 (含路径)。例如, 下面的这个 URL:

```
http://www.deitel.com/index.html
```

其协议为 http, 机器名称为 www.deitel.com, 对应的资源名称为 index.html。

Web 书店应用程序一旦被正确设置, 用户便可通过向 Web 浏览器中输入 URL 使其运行。相应操作将被转换为针对 Web 服务器 (如 Apache Tomcat) 的一个请求, 该服务器上驻留了 Web 书店应用程序。每当用户同 Web 浏览器进行交互以请求 Web 书店应用程序中的信息时, 该 Web 服务器会从信息层 (包含一个 Cloudscape 数据库) 中检索出所需要的信息。之后, 由 Web 书店应用程序所生成的 Web 内容将被返回至客户端的 Web 浏览器上, 即向用户显示检索到的内容。例如, 当用户选取书名并通过点击按钮的方式发送一个可查看所有图书名称的请求时, Web 书店应用程序所生成的 Web 内容中将包含一系列书名和一个按钮。而当用户发送一个只查看某特定图书信息的请求时, Web 书店应用程序所生成的 Web 内容中, 也只包含用户所选图书的相关信息 (如作者、价格、ISBN 编码、版本号、版权年限以及该书的简介)。我们将在下一节中学习安装 Apache Tomcat Web 服务器和 Web 书店应用程序。

为理解 Web 浏览器如何定位 Web 服务器上的文档, 有必要先来了解一些相关术语的含义:

1. 主机: 主机是一台用于存储及维护资源, 如 Web 页面、数据库及多媒体文件的计算机。在 Web 书店应用程序的例子中, 读者自己的计算机将作为主机使用, 这是因为与应用程序相关的所有资源, 包括 JSP, 数据库等都被存储在了这台计算机当中。
2. 域: 域代表 Internet 上的一组主机。每个域都拥有一个域名, 也叫 Web 地址, 它惟一地标识了 Internet 上某商业或组织计算机的位置。
3. 正式域名: 正式域名 (FQDN, fully qualified domain name), 也称机器名, 包括一个主机 (如代表 World Wide Web 的 www) 和一个域名 (含顶级域)。其中, 顶级域 (TLD, top-level domain) 是正式域名中位置最后且最为重要的一个组件。

为从 Web 服务器中请求文档, 用户必须知道该 Web 服务器软件所在的正式域名 (机器名)。例如, 要从 Deitel 的 Web 服务器中访问文档, 必须知道它的 FQDN www.deitel.com。FQDN www.deitel.com 指明其主机为 www, 顶级域为 com。在一个 FQDN 中, 相应的 TLD 常常还用来描述拥有该域的某个组织的类型。如 com TLD 通常代表的是一个商业部门, org TLD 通常代表的是一个非营利性组织, 而 edu TLD 则通常代表的是一个教育机构。除此之外, 每一个国家也都拥有自身的 TLD, 如 cn 代表中国, et 代表埃塞俄比亚, om 代表阿曼, us 则代表美国。

每个 FQDN 都将对应一个称为 IP (Internet Protocol, Internet 协议) 地址的数字地址, 这非常类似于某住宅楼的街区地址。人们通过使用某个街道地址来定位城市中的房屋或商业区, 同样的道理, 计算机是通过使用 IP 地址来查找 Internet 上其他计算机的。互联网上的每一台主机都拥有一个惟一的地址, 该地址是由四个以句号为分隔符的数所组成的, 如 63.110.43.82。域名系统 (DNS: Domain Name System) 服务器则是一台用于维护 FQDN 数据库及相应 IP 地址的计算机。将 FQDN 翻译成 IP 地址的过程称 DNS 查询。例如, 为访问 Deitel 的 Web 网站, 可在 Web 浏览器中输入 FQDN www.deitel.com。利用 DNS 查询, www.deitel.com 将翻译成 Deitel 的 Web 服务器的 IP 地址 (63.110.43.82)。在下一节中, 将学习安装 Apache Tomcat Web 服务器和完成后的 Web 书店应用程序。

自测题

1. _____ 是一台用于存储及维护相关资源的计算机。

- a) 主机 b) IP 地址 c) 域名 d) 域名系统
2. DNS 查询是指 _____。
- a) 将 IP 地址翻译成域名 b) 将正式域名翻译成 IP 地址
- c) 将正式域名翻译成主机名 d) 查询某个域名

答案：1) a 2) b

29.3 Apache Tomcat Web 服务器

Apache 组件中的 Tomcat，是一台用来创建并发布 Web 页面从而响应客户请求的 Web 服务器。在开始探试相关应用程序之前，必须首先安装 Tomcat 和 Web 书店应用程序。假定在安装 Tomcat 之前已安装好了 Java 2 软件开发包，如 J2SDK 1.4 所示（注意：如果读者一直都在运行和创建本书中的应用程序，则说明已安装了 J2SDK 1.4 版。如果已安装 Tomcat，可跳过下面的内容）。

安装 Tomcat 和 Web 书店应用程序

1. 安装 Tomcat 在随书附带的 CD-ROM 上找到名为 jakarta-tomcat-4.1.24-LE-jdk14.exe 的文件（位于 software\windows\tomcat 目录中），通过双击运行此安装程序，安装程序将一步一步地对 Tomcat 进行安装。
2. 确认 Java 开发包的位置 Tomcat 安装程序会自动定位到已安装的 J2SDK 目录。在首次出现的这个对话框中（如图 29.2 所示），点击 OK 按钮，确认 J2SDK 1.4 的安装目录。
3. 接受许可协议 为成功安装 Apache Tomcat，需接受相应的许可协议。当许可协议对话框出现时，应仔细阅读其中的条款，如果同意，请点击 I Agree 按钮（如图 29.3 所示）。

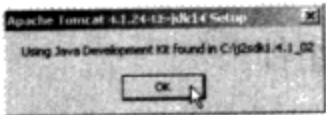


图 29.2 定位 Java 开发包

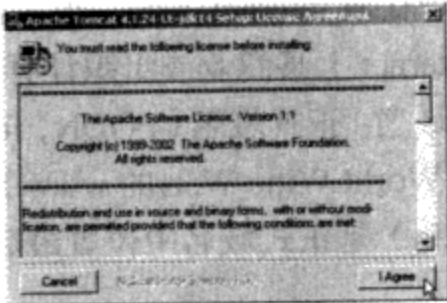


图 29.3 接受 Tomcat 许可协议

4. 接受默认安装选项 当出现提示用户选择安装类型时，点击 Next 按钮，接受所有默认选项（如图 29.4 所示）。

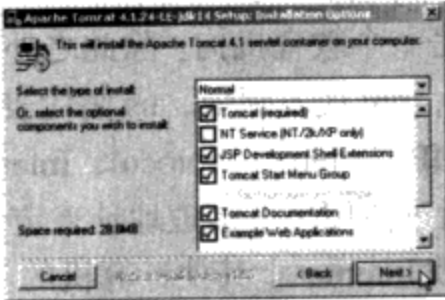


图 29.4 接受 Tomcat 默认安装选项

5. 指定安装目录 当出现提示用户选择安装目录时（如图 29.5 所示），点击 Install 按钮，将其安装在默认的目录中（如 C:\Program Files\Apache Group\Tomcat 4.1 所示）。
6. 监测安装过程 安装程序将把需要的文件复制到读者自己的计算机中（如图 29.6 所示）。
7. 设置管理员密码 Tomcat 中有一个内置的密码保护管理程序。当出现提示时，在 User Name 输入域中输入 admin，在 Password 输入域中同样也输入 admin，如图 29.7 所示（注意：实际可输入任何所希望的密码，但必须牢记）。为安全起见，最好是设置一个属于自己的密码。这里所输入的用户名和密码

2. 为了能成功安装 Web 书店应用程序，必须将它复制到 Tomcat 中的 _____ 目录下。
a) server b) work c) webapps d) 以上答案均不对

答案：1) b 2) c

29.4 探试 Web 书店应用程序

在随后的三个教程中，将构建一个 Web 应用程序，该应用程序将根据用户的请求显示出相关的图书信息。这个 Web 书店应用程序必须满足下面的需求：

应用程序需求分析

某书店员工同查询该书店在线图书销售信息的客户之间是通过接收E-mail的方式取得联系的。现在，要求为该员工开发一个 Web 应用程序，允许客户在线查询所有不同图书的信息。这些信息包括：图书的作者、封面照片、价格、ISBN 编码、版本号、版权年限及该书的一个简介。将创建的这个 Web 书店应使用 JavaServer Pages 实现其中间层，并利用 HTML 在客户层中实现该应用程序的 GUI。该应用程序 GUI 允许用户选择一个书名，之后可查看所选图书的相关信息。中间层将从信息层中检索出该图书的信息并把它显示在客户层上。信息层将用来维护所需的数据库，该数据库已由 Cloudscape 生成。使用 JDBC API 访问这个已提供给用户的数据库。

我们将以这个完成后的 Web 应用程序的探试作为开始，并在自己的本地计算机中测试该 Web 书店应用程序。绝大多数计算机都指定了一个特定主机名——localhost（通常对应着 IP 地址 127.0.0.1 所示）——作为在本地计算机中测试基于 Web 的应用程序（及其他网络化的应用程序）时使用。正如将在探试中所看到的那样，在访问 Web 书店应用程序的 URL 中，包含了一个可作为其主机名的 localhost。大多数情况下，还可直接指定自己计算机的名字而不是使用 localhost 作为主机名。（注意：假如读者运行的是微软 Windows 操作系统，计算机名的查看过程可通过选择 Start → Settings → Control Panel 从列表中双击系统，这时会出现一个系统特性对话框。如果是 Windows 2000，点取其的网络标识标签。在该标签内，可以看到完整的计算机名称：输出域中所显示的该台计算机的机器名。如果是 Windows XP，则在对话框中选择计算机名标签。同样，在该标签内可以看到完整的计算机名称：输出域中所显示的该台计算机的机器名。至于其他平台，请向指导教师咨询相关计算机的名称）。



探试基于 Web 的书店应用程序

1. 启动 Tomcat 为响应客户端请求，首先必须能让 Tomcat 运行起来。选择 Start → Programs → Apache Tomcat 4.1 → Start Tomcat 运行 Tomcat。此时，会出现一个类似图 29.11 的窗口。



图 29.11 Tomcat 服务器输出窗口

2. 运行应用程序 打开自己的 Web 浏览器，输入相应的 URL hppt://localhost:8080/bookstore/books.jsp。该操作会向 Tomcat Web 服务器请求 Web 书店应用程序中的一个页面：books.jsp。books.jsp 页面将显示一系列可以获得的图书列表（如图 29.12 所示）。尽管这个 GUI 列表组件看起来与读者曾经在桌面应用程序所使用的 JComboBox 组件相类似，而事实上该列表组件是一个 HTML 菜单控件，它是通过 HTML select 元素来定义的。我们将在教程 30 中学习如何使用 HTML 菜单控件。程序员通过添加各种 HTML 元素来定制 Web 页面。读者将在以后的教程中看到，一些 HTML 元素看上去的确同常见的那些 Java GUI 组件相类似。图 29.12 中的标注清楚地标明了各种不同的 HTML 元素，这些元素将在整个案例研究中得到阐述（教程 30 至教程 32）。



图 29.12 一个带有列表和按钮表单的 HTML 页面

在菜单控件中所显示的图书标题实际取自某个数据库。该数据库属于这个三层应用程序中的信息层。尽管该网页中只显示出了图书的标题，但数据库中其实还包含了其他一些与图书相关的信息，如作者、封面、价格、ISBN 编码、版本号、出版日期及简介等方面的信息。在教程 31 中，我们将考察这个应用程序的信息层，学习如何连接数据库并实现对数据的访问。

3. 选择图书 从图书列表中选择 C++ How To Program:Fourth Edition，然后点击 View Information 按钮。这时，会出现 bookInformation.jsp 页面（如图 29.13 所示）。此页面将显示所选图书的标题、作者及该书封面的照片。同时，还将显示该书的价格、ISBN 编码、版本号、出版日期及简介。

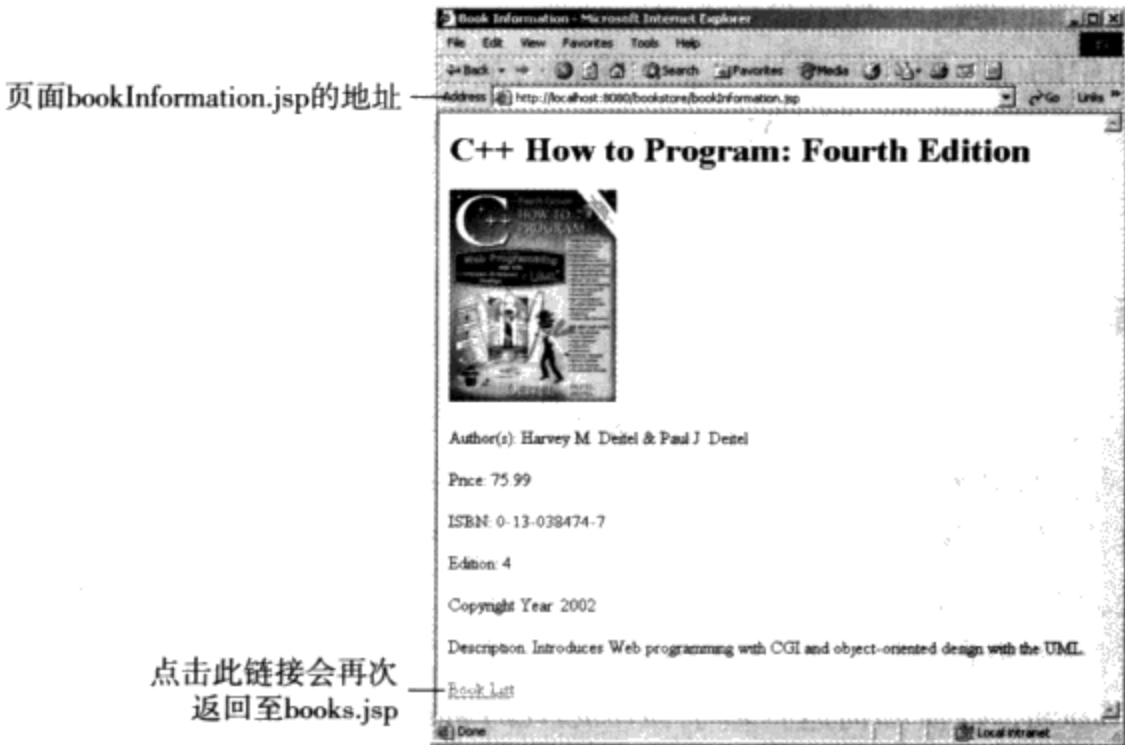


图 29.13 显示所选图书信息的页面

4. 返回至 books.jsp 在查看完该书信息以后，用户可决定是否愿意继续浏览其他图书的信息。因此，在该页面的底部含有一个 Book List 链接，点击它，浏览器会重新定位至含有相关图书标题列表表单的那个 books.jsp 页面上。

5. 关闭浏览器 点击浏览器窗口的关闭按钮关闭浏览器。

6. 从 Tomcat 中的 webapps 目录下删除已完成的应用程序 由于不允许在 webapps 目录中同时拥有两个同名的应用程序,因此,必须将上述这个已完成后的应用程序删除掉,才可能在随后的三个教程中开发属于自己的 Web 书店应用程序。点击 Start → Programs → Apache Tomcat 4.1 → Stop Tomcat 停止运行 Tomcat。之后,双击桌面上的“我的电脑”图标,定位至 Tomcat 的 webapps 文件夹,在 webapps 文件夹中,右击 bookstore 文件夹并选择快捷菜单中的 Delete,将 bookstore 文件夹删除(如图 29.14 所示)。

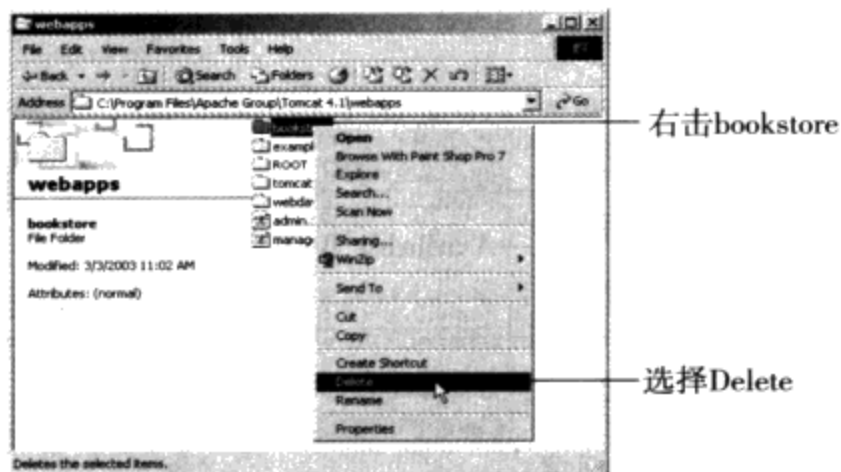


图 29.14 从 Tomcat 中的 webapps 目录下删除 bookstore 目录

注意,这个例子使用了一个应用程序中所有三层上的内容。为进行预览,信息层中包含了一个供 Web 应用程序检索信息的数据库。客户层是利用 Web 页面来表示的,用户在页面中查看并选择所需要的图书。中间层中的代码,用于控制当用户同请求数据库信息的 Web 页面发生交互时,可能出现的一些情况。

自测题

1. HTML 文档中的按钮是通过 _____ 来表示的。
a) HTML input 元素 b) HTML button 元素 c) HTML select 元素 d) 以上答案均不对
2. Web 页面中的项目列表可通过 _____ 来进行显示。
a) HTML input 元素 b) HTML button 元素 c) HTML select 元素 d) 以上答案都不对

答案: 1) a 2) c

29.5 小结

通过对在本教程中的学习,了解了三层 Web 应用程序中相关的组成单元。懂得了信息层(也称底层)可用来存储应用程序中的数据,这通常是用数据库实现的。我们还学习了用于显示应用程序用户接口的客户层(也称顶层),以及协调信息层和客户层之间交流的业务逻辑层(也称中间层)。之后,本教程还介绍了有关 Web 服务器的知识并安装了 Apache Tomcat Web 服务器。利用这个服务器,提供所需要的 Web 内容,如 JavaServer Pages。通过探试三层 Web 书店应用程序,学习了如何启动和停止 Tomcat,以及如何运行基于 JavaServer Pages 的应用程序。与此同时,还了解了有关创建 Web 页面的 HTML 元素。

在教程 30 中,将创建该应用程序的用户界面部分。通过设计 Web 页面,显示一个图书列表及相关图书的信息。在教程 31 中,还将学习如何在应用程序中使用数据库以及如何实现应用程序与数据库之间的连接。在教程 32 中,还将完成整个 Web 书店应用程序的创建。在最后的这个教程中,将包括完整 Web 书店应用程序的编写。

技术小结

安装 Tomcat

- 找到随书附带 CD-ROM 上的 jakarta-tomcat-4.1.24-LE-jdk14.exe，通过双击，启动安装程序。
- 按照所提示的指令完成整个安装过程。

启动 Tomcat

- 选择 Start → Programs → Apache Tomcat 4.1 → Start Tomcat。

停止 Tomcat

- 选择 Start → Programs → Apache Tomcat 4.1 → Stop Tomcat。

在计算机中设置一个 Web 应用程序

- 将该 Web 应用程序的文件夹放置在 Tomcat 中的 webapps 目录下（假如 Tomcat 安装在默认的目录下，则相应的 webapps 目录将位于 C:\Program Files\Apache Group\Tomcat 4.1\webapps）。

通过 Web 浏览器测试 Web 应用程序

- 打开一个 Web 浏览器
- 键入相应的 URL `http://localhost:8080/nameOfApplication/nameOfFirstPage`，其中，nameOfApplication 为某 Web 应用程序的目录名，nameOfFirstPage 为该应用程序运行时首个被载入的 Web 页面（比如一个 JSP 页面）。
- 通过按下 Enter 键运行应用程序。

关键术语

Apache Tomcat 来自 Apache Group 中一个免费使用且适用于多种平台的 Web 服务器。

底层 一个含多层应用程序数据的逻辑层（也称信息层或数据层）——通常是由数据库来实现的。

客户层 多层应用程序中的用户接口（也称顶层）。

数据层 一个包含多层应用程序数据的逻辑层（也称信息层或数据层）——通常是由数据库来实现的。

DNS 查询 将正式域名翻译成 IP 地址的过程。

域 代表 Internet 上的一组主机。

域名 也称 Web 地址，能够惟一标识 Internet 上某企业或组织的位置。

域名系统（DNS，Domain Name System）服务器 存储主机名及对应 IP 地址数据库的一台计算机。

正式域名（FQDN，fully qualified domain name） 组合了某域及其顶级域的一个主机名，从而提供了一个用户友好的方式来标识 Internet 上的某个站点。

主机 一台用于存储并维护相关资源，如 Web 页面、数据库、多媒体文件的计算机。

HTML 元素 属于 HTML 语言中的一部分，例如，h1，select 或 input，作为确定某 Web 页面的格式以及 Web 页面中所出现的 GUI 组件。

超文本标记语言（HTML，HyperText Markup Language） 一种可完全由 Web 浏览器解释的语言，通过使用一些表格和图像发布在线文档。

信息层 一个包含多层应用程序数据的逻辑层——通常是用数据库来实现的（也称信息层或数据层）。

IP（Internet Protocol: Internet 协议）地址 定位 Internet 上某台计算机的一个惟一的地址。

JavaServer Pages 使用 Java 及 HTML 开发动态 Web 页面的一项技术。

localhost 可标识计算机的一个主机名。

中间层 一个用于控制某多层应用程序客户层及其信息层之间交互的逻辑层。有时也称业务逻辑层。

多层应用程序 应用程序的功能被分散在多个独立的逻辑层上（有时也被引述为 n 层应用程序），这些独立的逻辑层可处在同一台机器上，或者是通过一个网络分布在多台不同的机器上。

多层应用程序的常见形式为三层结构 含应用程序用户界面的客户层；维护应用程序数据的信息层；以及用于控制应用程序客户（如 Web 浏览器）与信息层中的应用程序数据之间进行交互的业务逻辑层。

层 功能的逻辑分组。

顶级域(TLD, top-level domain) Web地址中描述拥有该域名的某个组织类型的部分。例如, com TLD 通常代表的是一个商业部门, org TLD 通常代表的是一个非营利性的组织, 而 edu TLD 则通常代表的是一个教育机构。

顶层 在一个多层应用程序中, 表示含有应用程序用户界面的逻辑层——通常是由 GUI 来实现的。

URL(统一资源定位器) 用于将浏览器定位至 Web 上某个资源的一个地址。

Web 地址 也称域名, 惟一标识了 Internet 上某个商业或组织的位置。

Web 应用程序 也称基于 Web 的应用程序, 可使用 JavaServer Pages (JSP) 技术创建 Web 内容。

Web 内容 可通过 Web 浏览器, 如 Internet Explorer 或 Netscape 进行浏览的数据。

Web 服务器 通过将 Web 页面及其他资源发送给客户以响应 Web 客户请求的特定软件。

习题

选择题

- 29.1 JavaServer pages 文件的扩展名为 ____。
- a) .html b) .jsp c) .jspx d) 以上答案均不对
- 29.2 ____ 应用程序可把功能分解到多个不同的逻辑层上。
- a) n 层 b) 多层 c) (a)和(b) d) 以上答案均不对
- 29.3 多层应用程序的所有层 ____。
- a) 必需位于同一台计算机上
- b) 必需位于多台不同的计算机上
- c) 可位于同一台计算机上或者是多台不同的计算机上
- d) 必须使客户层和中间层处于同一台计算机上, 而信息层则在另一台不同的计算机上。
- 29.4 客户层通过与 ____ 层进行交互来访问 ____ 层的信息。
- a) 中间, 信息 b) 信息, 中间 c) 信息, 底 d) 底, 信息
- 29.5 ____ 是利用 Web 浏览器向请求响应的客户提供所需资源的一个特定软件。
- a) 主机 b) 主机名 c) DNS 服务器 d) Web 服务器
- 29.6 ____ 被认为是能够将浏览器定位至 Web 上某个资源的一个地址。
- a) 中间层 b) ASPX 页面 c) URL d) 查询字符串
- 29.7 ____ 代表 Internet 上的一组 ____。
- a) 域, 主机 b) 主机, 域名 c) 主机名, 主机 d) 以上答案均不对
- 29.8 ____ 为一个 Web 服务器。
- a) Apache Tomcat b) localhost c) Java d) 以上答案均不对
- 29.9 ____ 是一个可代表本地计算机的特定主机名。
- a) localhost b) 本地 Web 服务器 c) 远程 Web 服务器 d) 以上答案均不对
- 29.10 ____ 层属于应用程序的用户界面。
- a) 中间 b) 客户 c) 底 d) 信息

练习题

- 29.11 (电话号码簿 Web 应用程序) 在随后的三个教程中, 将创建一个电话号码簿 Web 应用程序。此电话号码簿应用程序将作为教程 28 中所创建的那个电话号码簿应用程序的一个 Web 版本(注意: 该应用程序无需使用教程 28 中使用的 Java 语音 API)。电话号码簿 Web 应用程序由两个 JSP 页面组成, 其页面名称分别为 phoneBook 和 phoneNumber。phoneBook 页面将显示一列含有多个人员姓名的列表。这些人员姓名是从一个名为 phonebook 的数据库中检索出的。当选取某人姓名并按下 Get Number 按钮时, 客户浏览器会被重新定位至 phoneNumber JSP 页面, 而与所选人员姓名对应的电话号码将从数据库中被检索出来, 并显示在这个 phoneNumber JSP 页面中。针对这一练

习,读者只需将此 Web 应用程序中的相应组件 (phoneBook 和 phoneNumber JSP 页面、phonebook 数据库,以及执行特定功能的代码)组织到各自所属的不同逻辑层中,并确定各组件所对应的逻辑层。我们将在下一个教程中,构建此应用程序。

29.12 (美国各州知识检索 Web 应用程序)在随后的三个教程中,将创建一个美国各州知识检索 Web 应用程序。此应用程序允许用户预览美国一些州的文化知识。应用程序由两个 JSP 页面组成。第一个页面 (名为 states) 显示一系列含有 10 个不同州名的列表。这些州名被存储在一个名为 StateFacts 的数据库中。允许用户从该列表中选出一个州名并通过点击一个按钮,从数据库中检索出有关该州的一些信息。所检索出的信息将显示在另一个不同的 JSP 页面上 (名为 stateFacts)。stateFacts 页面中将显示一幅印有该州州旗的图片,以及列有该州的州政府、州花、州树和州鸟的列表 (均来自数据库)。这里,会向读者提供有关各州州旗的图片。针对这一练习,读者只需将此 Web 应用程序中的相应组件 (states 和 stateFacts JSP 页面、StateFacts 数据库以及执行特定功能的代码)组织到各自所属的不同逻辑层中,并确定各组件所对应的逻辑层。我们将在下一个教程中,构建此应用程序。

29.13 (道路标志预览 Web 应用程序)在随后的三个教程中,将学习创建一个道路标志预览 Web 应用程序。道路标志预览 Web 应用程序由两个 JSP 页面组成。该应用程序可向用户显示作为预览的道路标志,并同时为用户安排一个驾驶员的执照考试。在第一个页面 (名为 roadSigns) 中,将显示 15 个道路标志的图片,这些图片已为读者提供。该页面通过检索 RoadSigns 数据库,显示相应的图片信息。与此同时,该页面中还将包含两个输入区域 (其外观类似于 JTextField) 及一个按钮,作为用户注册驾驶员执照考试输入信息时使用。当点击 Register 按钮时,第二个页面 (roadTestRegistered) 将显示用以确认用户已注册驾驶员执照考试的信息。针对这一练习,读者只需将此 Web 应用程序中的相应组件 (roadSigns 和 roadTestRegistered JSP 页面、RoadSigns 数据库以及执行特定功能的代码)组织到各自所属的不同逻辑层中,并确定各组件所对应的逻辑层。我们将在下一个教程中,构建此应用程序。

教程 30 Web 书店应用程序：客户层

HTML 简介



教学目标

在本教程中，读者将学到以下内容：

- 创建一个 JSP Web 应用程序
- JSP 页面的设计与创建
- HTML 表单控制及其他 HTML 元素
- 运用上述技术实现基于 Web 三层应用程序的客户层

在本教程中，读者将使用超文本标记语言（HyperText Markup Language，HTML）创建客户层（也称顶层），即三层 Web 书店应用程序的用户界面。HTML 是一种标记语言，用来指定 Web 浏览器，如 Internet Explorer（Microsoft 公司）和 Netscape（Netscape 公司）中文本的显示格式。我们先从创建基于 Web 应用程序的 JavaServer Pages（JSP）开始，然后使用 HTML 创建该应用程序的 GUI。

30.1 分析 Web 书店应用程序

在教程 29 中，我们曾探试过完成后的三层 Web 书店应用程序。下面，将对应用程序中的组件进行分析。以下伪代码描述了 Web 书店应用程序的基本操作：


当访问 books.jsp 页面时
 从数据库中检索书名
 将书名显示在 HTML 菜单中

当用户从菜单中选择某书名并点击 View Information（提交）按钮时
 根据所选书名请求 bookInformation.jsp 页面

当从 books.jsp 中请求 bookInformation.jsp 页面时
 从数据库中返回所选书的信息
 在 bookInformation.jsp 页面中格式化所返回的信息
 将请求结果返还客户端浏览器

当用户在 bookInformation.jsp 页面上点击 Book List 链接时
 请求 books.jsp 页面

现在我们已探试了 Web 书店应用程序并研究了它的伪代码表示。下面，将使用一张 ACE 表，把伪代码转换为相应的 JSP 页面。图 30.1 中列出了该应用程序中相应的操作、组件以及事件，帮助读者最终完成属于自己的应用程序。

	操作	组件 / 类 / HTML 元素	事件 / 方法
	从数据库中检索书名	books.jsp Statement	用户请求 books.jsp 页面

(续表)

操作	组件/类/HTML 元素	事件/方法
将书名显示在 HTML 菜单中	ResultSet HTML select 元素 HTML option 元素	用户点击“提交”按钮
根据所选书名请求在 bookInformation.jsp 页面中	类型为 submit 的 HTML input 元素 bookInformation.jsp	用户请求 bookInformation.jsp 页面
从数据库中返回所选书的信息 在 bookInformation.jsp 页面 中格式化所返回的信息	Statement ResultSet HTML p(paragraph)元素 HTML img 元素	
将请求结果返还客户端浏览器	HTML a(anchor)元素	用户点击超链接
请求 books.jsp 页面	books.jsp	

图 30.1 基于 Web 书店应用程序的 ACE 表

30.2 创建 JavaServer Pages

在教程 29 中，读者对 Apache Tomcat Web 服务器和基于 Web 的三层应用程序等概念已有了初步了解。下面，我们将开始创建这个 Web 书店应用程序。用户通过此 Web 应用程序浏览与所选图书有关的信息。查看完相关图书信息以后，用户可再次返回到原先包含书目信息的页面并重新选择其他图书。读者将在“创建 books.jsp 页面”和“创建 bookInformation.jsp 页面”两节中开发 Web 书店应用程序的 JSP 页面。

在某些方面，JSP 类似于标准的 HTML 文档。事实上，JSP 中既包含 HTML 标记也包含 JSP 代码。在本教程中，初始时创建的 books.jsp 和 bookInformation.jsp 页面只包含 HTML 部分。在教程 31 中，读者将给这两个 JSP 页面添加适当的 JSP 代码(注意：本书附带的 CD-ROM 中提供了从“Internet and World Wide Web How to Program”(第 2 版)中节选出的两章关于 HTML 的教程。如果想阅读这两个 PDF 格式的文件，在 Web 浏览器中打开 additional-references.htm 页面即可(位于随书附带 CD-ROM 中 html/additional-references 的目录下)。

30.3 创建 books.jsp 页面

Web 书店应用程序包含两个 JSP 页面。首先，创建用来显示书目信息的 books.jsp 页面。

创建 books.jsp 页面

- 1. 将模板复制到工作目录中 将 C:\Examples\Tutorial30\TemplateApplication\bookstore 目录复制到 C:\SimplyJava 目录中。
- 2. 打开 books.jsp 模板文件 在文本编辑器中打开 books.jsp 模板文件。
- 3. 设置该 JSP 的标题 将图 30.2 中第 10 行至第 11 行添加到 JSP 中。第 10 行是一个 HTML 注释，表示下一行为 JSP 标题。HTML 注释起始于“<!--”而终止于“-->”，且支持多行文本。当用户在 Web 浏览器中请求 JSP 文档时，注释行不会让 Web 服务器执行任何操作。HTML 标记包含表示文档内容的文本和文档结构与显示格式的一些元素。HTML 中的每个元素都有起始标记和结束标记。起始标记是由

位于一对尖括号内的元素名组成的（例如<title>）。结束标记是由位于一对尖括号内且前有“/”的元素名组成的（例如</title>）。第 11 行通过指定 title 元素将 JSP 标题设置为 Book List。title 元素用于设置浏览器窗口标题栏内显示的文本。图 30.8 显示了使用 title 元素后浏览器中的效果。title 元素总是出现在 head 元素中（如图 30.2 中第 8 行和第 12 行）。head 元素指定文档的不同格式，如文档标题等。

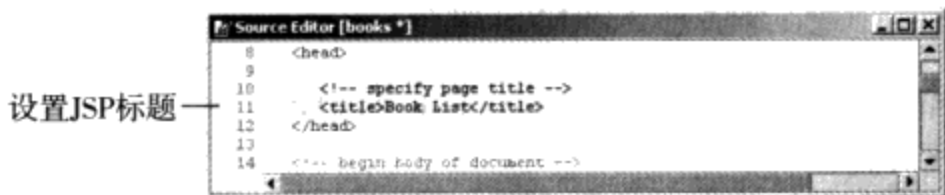


图 30.2 为 JSP 添加标题

4. 为 JSP 的 body 部分添加 header 元素 body 元素（位于图 30.3 中第 15 行和第 18 行）内包含的其他 HTML 元素会在 JSP 载入浏览器时显示到 Web 浏览器窗口中。body 元素可包含文档内容，如文本、图像、按钮等。HTML 通过提供不同层次的 header 元素来表示信息的相对重要程度。将图 30.3 中第 16 行的 header 元素插入 JSP，显示文本 Available Books。header 元素 h1（参见第 16 行）为该元素的最高级（即最大字体），用来在 Web 浏览器中显示大号字体。图 30.8 显示了使用 h1 元素后浏览器中的效果。

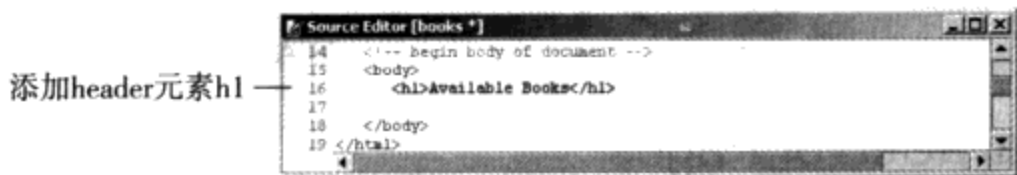


图 30.3 在 JSP 中添加 header 元素

5. 创建表单 用户在 books.jsp 菜单中选择某书标题，然后点击一个按钮浏览该书的信息。HTML 提供了一种称为表单的机制收集用户输入的数据。添加图 30.4 中第 18 行至第 21 行，在 JSP 中创建一个表单。

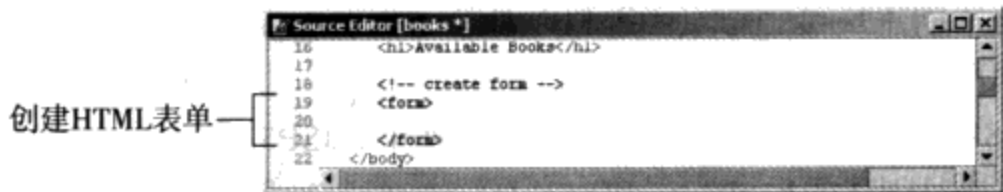


图 30.4 在 JSP 中创建表单

6. 创建文本段 将图 30.5 中第 21 行至第 22 行添加到 JSP 中 form 元素的内部。位于第 21 行至第 22 行中的段落标记（<p>和</p>）用于界定显示在 p（段落）元素内的文本。所有<p>和</p>标记内的文本都将构成一个文本段。浏览器通常会在每一个文本段的上下位置处设置一空白行。图 30.8 显示了使用段落元素后浏览器中的效果。

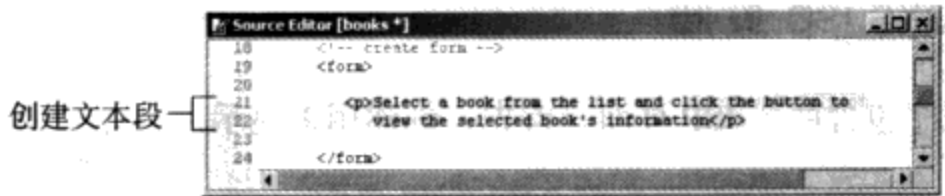


图 30.5 在 JSP 中创建文本段

7. 创建菜单 将图 30.6 中第 24 行至第 27 行添加到 JSP 中 form 元素的内部。select 元素（参见第 25 行至第 27 行）提供了一个下拉列表，也称 HTML 菜单控件，用户可从中选择一个选项（注意：在 Java 中通常将 GUI 元素称为“组件”，但在 HTML 中，则常被称为“控件”）。图 30.8 显示了使用 select 元素后浏览器中的效果。可以看到，下拉列表的尺寸（宽度）较小，这是因为读者尚未在其中添加选项。另外，name 属性用于标识下拉列表，这同 Java 代码中的变量名类似。此时，使用 select 元素并没有显示任何书名。在随后的两个教程中，我们将根据数据库中存储的书名检索其信息，然后写入这个 select 元素中。

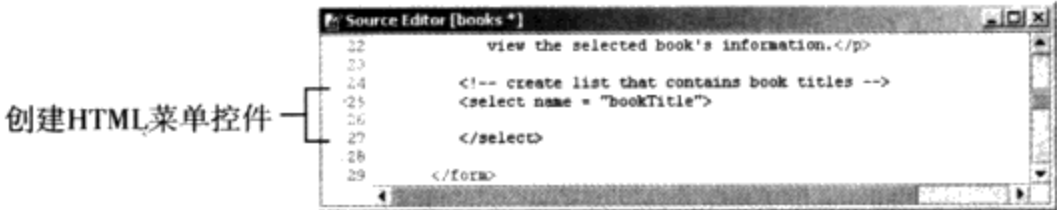


图 30.6 在 JSP 网页中创建一个空 select 元素

8. 添加按钮 将图 30.7 中第 29 行至第 30 行添加到 JSP 中，插入一个按钮控件。第 30 行定义了一个 input 元素，其 type 属性被设置为 "submit"，表示该 input 元素属于一个 "submit" 按钮。通过 input 元素可在 JSP 中添加各种不同的输入控件（如文本输入控件、按钮控件）。在这个 JSP 中，位于第 30 行的 input 元素定义了一个按钮控件。在练习 30.13 中，还将使用 input 元素为 JSP 添加一个文本输入控件。图 30.8 显示了使用 "submit" 按钮后浏览器中的效果。当点击 "submit" 按钮时，浏览器就会将表单中的数据发送到 Web 服务器中以等候处理。value 属性用于设置按钮上显示的文本内容。如果没有指定 value 属性，那么默认的值是 Submit。

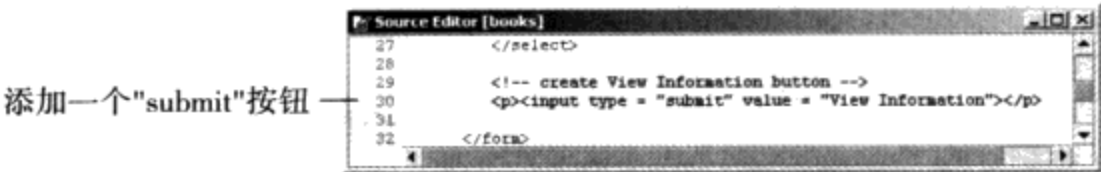


图 30.7 为 JSP 添加 "submit" 按钮

- 9. 保存应用程序 保存修改后的源代码文件。
- 10. 将 Web 书店应用程序复制到 Tomcat 中 webapps 目录下 为运行已得到更新的 Web 书店应用程序，需要将 C:\SimplyJava\bookstore 目录复制到 Tomcat 中 webapps 目录下。webapps 目录的默认安装目录为 C:\Program Files\Apache Group\Tomcat4.1\webapps。
- 11. 运行 Tomcat 点击 Start → Programs → Apache Tomcat 4.1 → Start Tomcat。
- 12. 运行应用程序 打开一个 Web 浏览器，输入 URL 地址 hppt://localhost:8080/bookstore/books.jsp。图 30.8 显示了更新后的 JSP。注意，HTML 菜单控件中并未包含任何书名信息，这是因为我们尚未与数据库建立连接，将在随后的两个教程中添加此项功能。点击 View Information 按钮，可以看到，并没有产生任何结果。因为当前仅仅只是定义了该页面的外观显示，所以点击 View Information 按钮不会进入 bookInformation.jsp。我们将在教程 32 中添加此项功能。



图 30.8 更新后的 JSP（含 HTML 表单）

- 13. 关闭浏览器 点击浏览器窗口的关闭按钮关闭浏览器。
- 14. 停止 Tomcat 选取 Start → Programs → Apache Tomcat 4.1 → Stop Tomcat 停止 Tomcat Web 服务器。

自测题

- 1. 使用 _____ 属性可以改变 "submit" 按钮上显示的文本。
a) text b) name c) value d) 以上答案都不对
- 2. _____ 元素可定义一个 HTML 菜单控件（下拉列表）。
a) input b) select c) h1 d) title

答案：1) c 2) b

30.4 创建 bookInformation.jsp 页面

在下面的内容中，我们将创建 bookInformation.jsp 页面，该页面用来显示列表中所选图书的信息。

创建 bookInformation.jsp 页面

- 1. 打开 bookInformation.jsp 模板文件 在文本编辑器中打开模板文件 bookInformation.jsp（位于工作目录 C:\SimplyJava\bookstore 中）。
- 2. 设置 JSP 标题 将图 30.9 中第 10 行至第 11 行添加到 JSP 中。第 11 行将此 JSP 的标题设置为 Book Information，此标题将出现在 Web 浏览器的标题栏内。图 30.19 显示了使用 title 元素后 Web 浏览器中的效果。

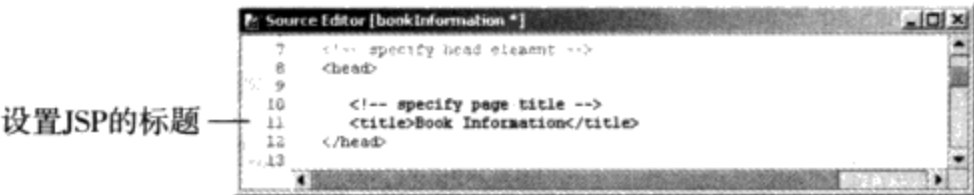


图 30.9 为 bookInformation.jsp 添加标题

- 3. 为 JSP 添加 header 元素 将图 30.10 中第 17 行至第 18 行添加到 JSP 中。第 18 行创建一个空的 header 元素 h1，用于显示所选书名。我们将在教程 32 中指定其中的文本。

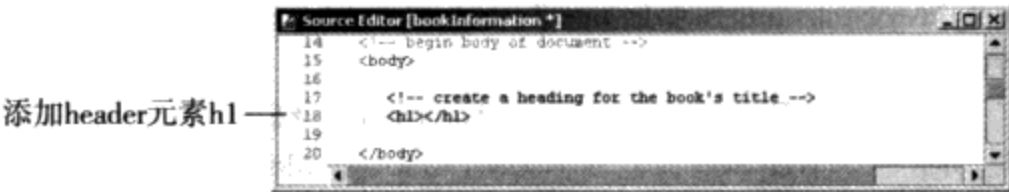


图 30.10 添加指定书名的 header 元素

- 4. 在 JSP 中插入图片 将图 30.11 中第 20 行至第 21 行添加到 JSP 中。第 21 行使用 img 元素在 JSP 中插入一张图片，作为显示所选书籍的封面。图片文件的位置由 img 元素的 src 属性具体指定。此时该值为空，我们将在教程 32 中具体进行设定。每个 img 元素都拥有一个 alt 属性——当浏览器无法显示图片时（例如 Web 服务器找不到图片文件），浏览器就会显示 alt 属性所指定的值（一个字符串）。在本例中，src 属性和 alt 属性的值均为空字符串。我们将在教程 32 中改变这些值。图 30.19 显示了使用 img 元素后浏览器中的效果。同 input 元素一样，img 元素也是一个空白元素，它的标记之间没有文本。

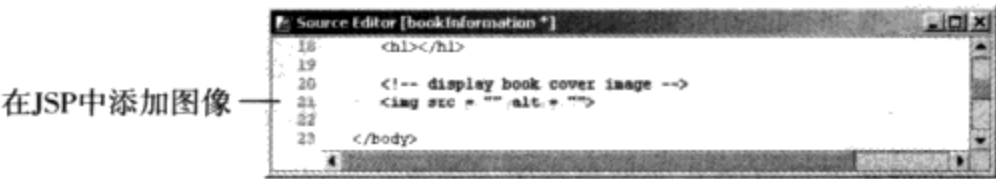


图 30.11 添加用于指定图书封面的 img 元素

- 5. 创建作者信息文本段 将图 30.12 中第 23 行至第 24 行代码添加到 JSP 中。第 24 行创建显示其内容为 "Author(s):" 的文本段。在教程 32 中，将根据数据库中检索出的作者信息来完成这个 p(paragraph)元素。图 30.19 显示了使用 paragraph 元素后浏览器中的效果。
- 6. 创建价格文本段 将图 30.13 中第 26 行至第 27 行添加到 JSP 中。第 27 行创建显示其内容为 "Price:" 的文本段。在教程 32 中，将通过数据库中检索出的价格信息来完成这个 p(paragraph)元素。图 30.19 显示了使用 paragraph 元素后浏览器中的效果。

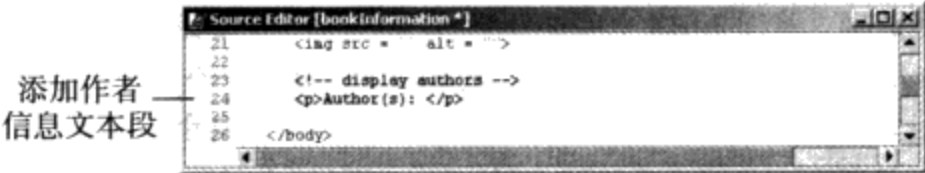


图 30.12 添加作者信息文本段

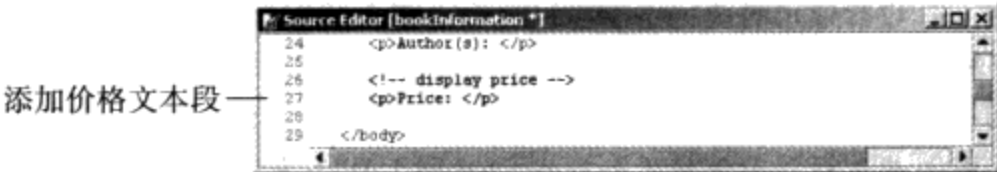


图 30.13 添加价格文本段

7. 创建 ISBN 文本段 将图 30.14 中第 29 行至第 30 行添加到 JSP。第 27 行创建显示其内容为 "ISBN:" 的文本段。在教程 32 中，将根据数据库中检索出的 ISBN 编号信息来完成这个 p(paragraph)元素。图 30.19 显示了使用 paragraph 元素后浏览器中的效果。

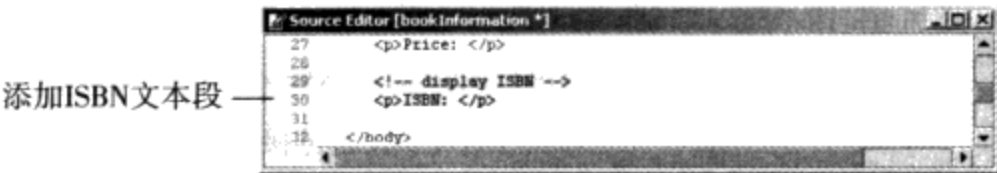


图 30.14 添加 ISBN 文本段

8. 创建版本文本段 将图 30.15 中第 32 行至第 33 行添加到 JSP 中。第 33 行创建显示其内容为 "Edition:" 的文本段。在教程 32 中，将根据数据库中检索出的版本信息来完成这个 p(paragraph)元素。图 30.19 显示了使用 paragraph 元素后浏览器中的效果。

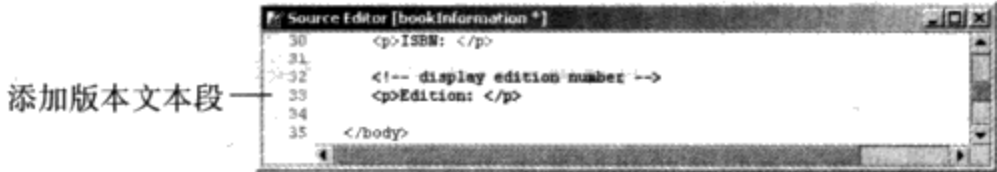


图 30.15 添加版本文本段

9. 创建出版日期段 将图 30.16 中第 35 行至第 36 行添加到 JSP 中。第 36 行创建显示其内容为 "Copyright Year:" 的文本段。在教程 32 中，将根据数据库中检索出的版权年限信息来完成这个 p(paragraph)元素。图 30.19 显示了使用 paragraph 元素后浏览器中的效果。

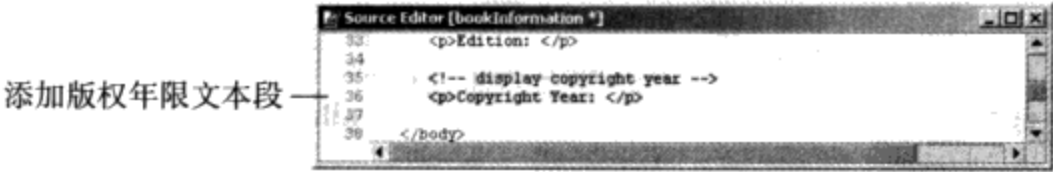


图 30.16 添加版权年限文本段

10. 创建描述文本段 将图 30.17 中第 38 行至第 39 行添加到 JSP 中。第 39 行创建显示其内容为 "Description:" 的文本段。在教程 32 中，将根据数据库中检索出的描述信息来完成这个 p(paragraph)元素。图 30.19 显示了使用 paragraph 元素后浏览器中的效果。

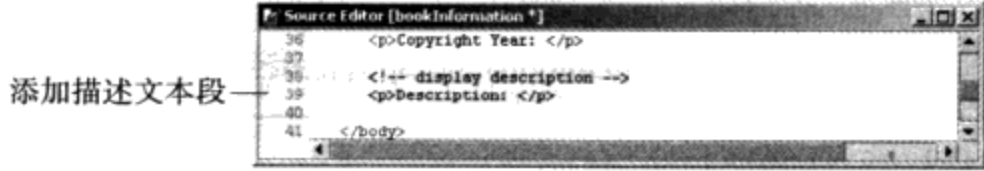


图 30.17 添加描述文本段

11. **建立 Book List 超链接** 将图 30.18 中第 41 行至第 42 行添加到 JSP 中。HTML 最重要的一项功能就是超链接 (hyperlink), 利用超链接可以访问 (或链接) 另外的一个资源, 如 HTML 文档或 JSP。利用 a (锚) 元素可以创建超链接。图 30.19 显示了使用 a 元素后浏览器中的效果。第 42 行定义的超链接能够将文本 Book List 链接到由 href (超链接访问, hyperlink reference) 属性所指定的 URL 上。href 属性用于指定某链接源的位置, 它可以是 Web 页面、文件或者 E-mail 地址。这里所给出的锚元素将链接至 books.jsp。注意, 第 42 行使用的是一个相对 URL 作为链接源的位置。相对 URL 不包含源协议 (如 http)、源域名或源 IP 地址。相对 URL 只包含源路径。在第 42 行, 相对 URL 为 "books.jsp", 即所访问的 books.jsp 页面同当前 JSP 页面 (bookInformation.jsp) 位于同一个目录内。相对 URL 只能是在所链接的 Web 源属于同一个 Web 应用程序时使用 (如 books.jsp 属于 Web 书店应用程序)。如果所连接的 Web 资源不属于同一个 Web 应用程序 (如 www.deitel.com), 那么 href 属性的值就必须为一个完整的 URL (如 www.deitel.com)。超链接的默认操作是访问由 href 属性所指定的 Web 资源。因此, 当用户点击图 30.19 中的 Book List 超链接时, 便会返回 books.jsp 页面并显示在客户端 Web 浏览器中。

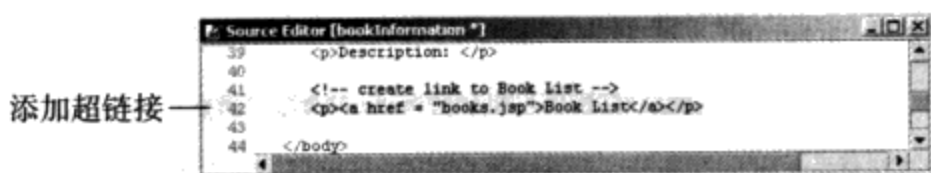


图 30.18 添加 Book List 超链接

12. **保存应用程序** 保存修改后的源代码文件。
13. **将 bookInformation.jsp 复制到 Tomcat 的 webapps 目录中** 将完成后的 bookInformation.jsp 文件从 C:\SimplyJava\bookstore 中复制到 C:\Program Files\Apache Group\Tomcat 4.1\webapps\bookstore 目录下。
14. **运行 Tomcat** 选取 Start → Programs → Apache Tomcat 4.1 → Start Tomcat。
15. **测试应用程序** 打开一个 Web 浏览器并在 URL 地址栏中输入 http://localhost:8080/bookstore/bookInformation.jsp。图 30.19 显示了更新后的 JSP。注意, 文本段中并不包含任何与图书有关的信息, 因为此时尚未建立数据库连接。我们将在随后的两个教程中实现这一功能。点击 Book List 超链接, 浏览器会载入 books.jsp 页面。
16. **关闭浏览器** 点击浏览器窗口的关闭按钮关闭浏览器。
17. **停止 Tomcat** 选取 Start → Programs → Apache Tomcat 4.1 → Stop Tomcat 停止 Tomcat 服务器。

图 30.20 给出了 books.jsp 的源代码。图 30.21 给出了 bookInformation.jsp 的源代码。本教程中, 凡需要添加、查看或者修改的代码, 均在图中进行了突出显示。

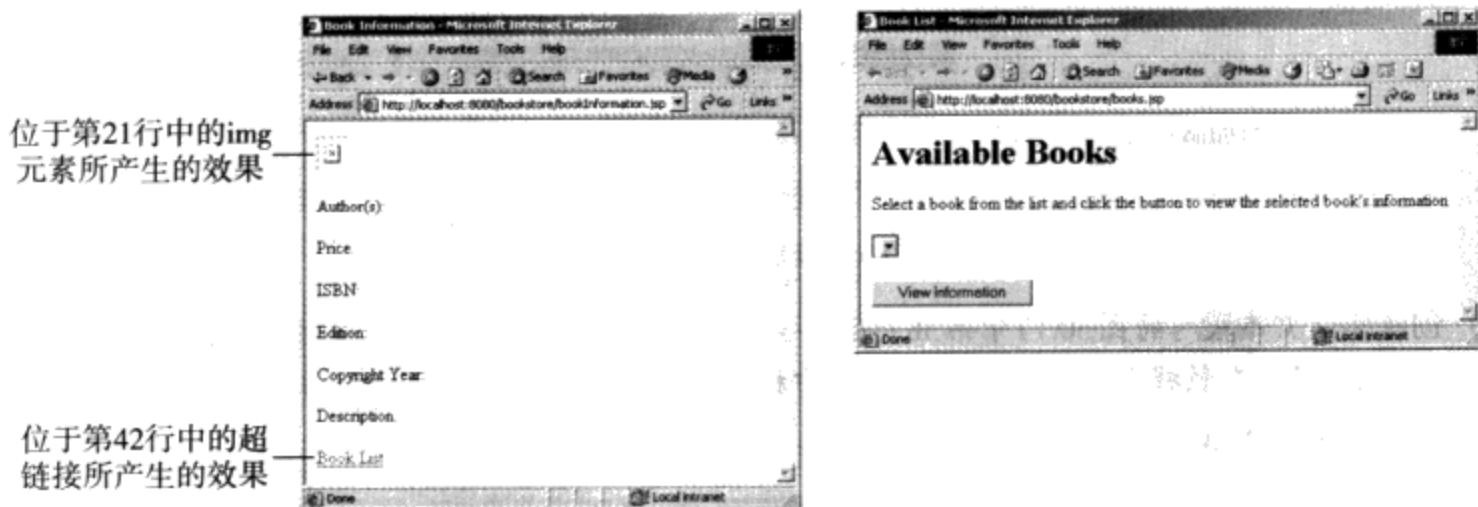


图 30.19 显示 bookInformation.jsp 并链接至 books.jsp

```
1 <!-- Tutorial 30: books.jsp -->
2 <!-- Displays a form. -->
3
```



```

4 <!-- begin HTML document -->
5 <html>
6
7   <!-- specify HTML head element -->
8   <head>
9
10      <!-- specify page title -->
11      <title>Book List</title>
12  </head>
13
14  <!-- begin body of document -->
15  <body>
16      <h1>Available Books </h1>
17
18      <!-- create form -->
19      <form>
20
21          <p>Select a book from the list and click the button to
22              view the selected book's information</p>
23
24          <!-- create list that contains book titles -->
25          <select name = "bookTitle">
26
27              </select>
28
29          <!-- create View Information button -->
30          <p><input type = "submit" value = "View Information"></p>
31
32      </form>
33  </body>
34 </html>

```

设置JSP标题

添加header元素h1

添加一个含文本段、下拉列表、“submit”按钮的HTML表单

图 30.20 books.jsp 的源代码

```

1 <!-- Tutorial 30: bookInformation.jsp -->
2 <!-- Displays book information. -->
3
4 <!-- begin HTML document -->
5 <html>
6
7   <!-- specify head element -->
8   <head>
9
10      <!-- specify page title -->
11      <title>Book Information</title>
12  </head>
13
14  <!-- begin body of document -->
15  <body>
16
17      <!-- create a heading for the book's title -->
18      <h1></h1>
19
20      <!-- display book cover image -->
21      <img src = "" alt = "">
22
23      <!-- display authors -->
24      <p>Author(s): </p>
25

```

设置JSP标题

添加一个空白header元素h1

添加图片

添加作者信息文本段

```
26      <!-- display price -->
27      <p>Price: </p>
28
29      <!-- display ISBN -->
30      <p>ISBN: </p>
31
32      <!-- display edition number -->
33      <p>Edition: </p>
34
35      <!-- display copyright year -->
36      <p>Copyright Year: </p>
37
38      <!-- display description -->
39      <p>Description: </p>
40
41      <!-- create link to Book List -->
42      <p><a href = "books.jsp"> Book List</a></p>
43
44      </body>
45 </html>
```

图 30.21 bookInformation.jsp 的源代码

自测题

1. img 元素的 _____ 属性可用于定位图片文件的位置。
a) location b) source c) src d) 以上答案都不对
2. a (锚) 元素的 _____ 属性能够指定某链接源的位置。
a) href b) location c) reference d) link

答案: 1) c 2) a

30.5 小结

在本教程中,我们创建了三层 Web 书店应用程序中的 JavaServer Pages (JSP) 页面。学习了如何为 JSP 添加 HTML 控件及相关的元素。在本教程中用到的 HTML 控件包括: 菜单控件 (通过 select 元素设定)、"submit" 按钮 (将 input 元素的 type 属性设定为 "submit")。除此以外,还学习了其他一些 HTML 元素,包括: head, body, form, header (h1), 文本段 (p), 图片 (img), 超链接 (a), 以及各种不同元素属性的用法,如: img 元素的 alt 属性和 src 属性, select 元素的 name 属性, input 元素的 type 属性和 value 属性, a(anchor) 元素的 href 属性。

在下一个教程中,将访问应用程序的数据库 (也称信息层), 该层包含了书店内图书的相关信息。读者将使用必要的 SQL 语句检索数据库中的图书信息。利用 Connection 对象和 Statement 对象执行数据库的连接和处理操作。在创建完数据库中的组件以后,将在教程 32 中构建此书店应用程序的中间层,为 JSP 提供服务。

技术小结

设置 JSP 的标题

- 使用 HTML 的 title 元素设置浏览器窗口标题栏内显示的标题。

为 JSP 添加 header 元素

- 使用 HTML 的 h1 元素添加标题。

创建 HTML 表单

- 使用 form 元素收集用户输入。

在表单中添加 HTML 菜单控件

- 使用 HTML 的 select 元素为表单添加一个下拉列表。

在表单中添加 "submit" 按钮

- 使用 type 属性值为 "submit" 的 input 元素创建 "submit" 按钮。利用 value 属性指定显示在按钮上的文本。

在 JSP 中显示图片

- 使用 HTML 的 img 元素在 JSP 中添加图片。利用 src 属性指定图片文件的位置，若浏览器无法定位该图片文件时，可以使用 alt 属性给出一条文本显示。

在 JSP 中添加超链接

- 使用 HTML 的 a(anchor)元素添加超链接，从而链接到另外的一个资源上，如一个 JSP 或 HTML 文档。使用 href 属性指定链接源的位置。在 JSP 中可通过设置文本实现超链接功能。

关键术语

HTML 中的 a(anchor)元素 用于创建超链接。

HTML img 元素的 alt 属性 当浏览器无法定位图片时，指定要显示的文本。

HTML 中的 body 属性 包含 Web 浏览器窗口中用于显示的 HTML 元素。

HTML 中的注释 起始于 "<!--" 而终止于 "-->", HTML 注释用来描述 HTML 文档或 JSP, 它支持多行文本。当浏览器显示 HTML 时, 会忽略其中的注释。

HTML 中的元素 用于指定文档结构的标记。

空白元素 在起始标记和结束标记之间不包含任何文本的元素。

HTML 中的结束标记 元素名前有一个 "/" 的尖括号对, 表示某个 HTML 元素的结束。

HTML 中的 form 元素 用于收集用户输入。

HTML 中的 h1 元素 最高一级的 header 元素。通常浏览器会将 h1 元素中的内容以大号字体显示。

HTML 中的 head 元素 设定文档的不同结构, 如标题, 等等。

HTML 中的 header 元素 用来指定信息之间的相对重要程度。

HTML 标记 包含表示文档内容的文本以及指定文档结构的元素。

HTML a(anchor)元素的 href (超链接资源) 属性 指定链接源的位置。

HTML 中的菜单控件 利用 HTML 的 select 元素创建的一个下拉列表。

HTML 中的超链接 用于访问其他源文件, 如 HTML 文档或 JSP。

超文本标记语言 (HTML) 一种用于指明 Web 浏览器中的文本及图片显示格式的标记语言, Web 浏览器包括 Internet Explorer 或 Netscape。

HTML 中的 img 元素 在 HTML 文档中插入图片。

HTML 中的 input 元素 用于创建 "submit" 按钮。

HTML select 元素的 name 属性 用于标识下拉列表的菜单控件。所指定的标识符同 Java 中的变量名类似。

HTML 中的 p(paragraph)元素 为 JSP 或 HTML 文档添加一个文本段。

相对 URL 相对 URL 不包含源协议 (如 http)、域名或 IP 地址。相对 URL 只包含源路径。

HTML 中的 select 元素 用于创建 HTML 菜单控件 (也称下拉列表)。

HTML img 元素的 src 属性 指定图片文件的位置。

HTML 的起始标记 位于一对尖括号内的元素名, 表示一个 HTML 元素的开始。

"submit" 按钮 通过将 HTML 中 input 元素的 type 属性设置为 "submit" 而创建的一个按钮。

HTML 中的 title 元素 设置浏览器标题栏内所显示的标题。

HTML input 元素的 type 属性 指定 input 元素的类型。将 type 设置为 "submit" 表示此 input 元素为一个 "submit" 按钮。

HTML input 元素的 value 属性 设置显示在 "submit" 按钮上的文本。

HTML 参考索引

a(anchor) 该元素作为在 HTML 文档中添加超链接。

- 属性

href 指定链接源的位置。

body 此元素包含了 Web 浏览器窗口中用于显示的 HTML 元素。

form 此元素用于在 HTML 文档中插入一个收集用户输入的表单。

h1 最高一级的 header 元素，其内容将在浏览器中以大号字进行显示。

head 此元素用于指定文档结构，如文档标题。

img 此元素用于在 HTML 文档中插入一张图片。

- 属性

alt 当浏览器无法定位图片时，指定一段用于显示的文本。

src 指定图片文件的位置。

input 此元素用于创建 "submit" 按钮。

- 属性

type 指定 input 元素的类型。将 type 设置为 "submit"，表示 input 元素是一个 "submit" 按钮。

value 设置 "submit" 按钮上所显示的文本。

p (paragraph) 此元素用来在 JSP 或 HTML 文档中添加文本段。

select 此元素用于创建 HTML 菜单控件。

- 属性

name 用于标识下拉列表菜单控件。所使用的标识名与 Java 中的变量名类似。

title 此元素用于设置显示在浏览器窗口标题栏内的文本。

习题

选择题

30.1 HTML input 元素的 _____ 属性可指定 "submit" 按钮上所显示的文本。

- a) text b) name c) value d) button

30.2 使用 _____ 元素可将 HTML 菜单控件添加到 JSP 中。

- a) select b) menu c) input d) 以上答案都不对

30.3 HTML 中的 _____ 元素可为 JSP 添加超链接功能。

- a) p b) link c) h1 d) a

30.4 title 元素用于设置 _____。

- a) 显示在按钮上的文本 b) JSP 页面中的第一行显示
c) 浏览器窗口标题栏内的标题 d) 以上答案都不对

30.5 HTML img 元素的 src 属性 _____。

- a) 指定图片的类型 b) 指定图片文件的位置
c) 指定浏览器无法定位图片时的文本显示 d) 指定图片的大小

- 30.6 HTML 的 _____ 元素包含浏览器窗口中显示的 HTML 元素。
a) input b) head c) body d) title
- 30.7 _____ 元素用来创建 HTML 表单。
a) form b) htmlForm c) h1 d) 以上答案都不对
- 30.8 HTML 注释起始于 _____ 终止于 _____。
a) /*, */ b) <!--,--> c) <--,--> d) <!--,--!>
- 30.9 HTML 中的 _____ 提供了一种从 Web 页面中收集用户输入的机制。
a) 标记 b) 表单 c) 元素 d) 注释
- 30.10 使用 _____ 可链接到其他资源。
a) 超链接 b) 按钮控件 c) 菜单控件 d) 以上答案都不对

练习题

- 30.11 (电话号码簿应用程序: GUI) 创建电话号码簿应用程序的用户界面。有关此应用程序的详细介绍请参见习题 29.11。为应用程序设计的两个页面如图 30.22 所示。我们还将随后的两个教程中进一步完善该应用程序, 所以, 此时尚不能从数据库中读取相关名称及数量信息。为帮助读者更好地理解本应用程序, 图 30.23 给出了最终完成设计以后的屏幕效果图。



图 30.22 电话号码簿应用程序的用户界面

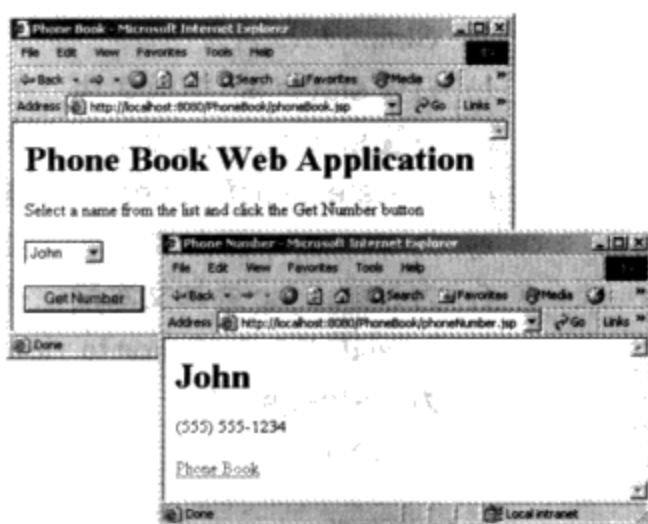


图 30.23 完成后的电话号码簿应用程序

- 将模板复制到工作目录中 将 C:\Examples\Tutorial30\Exercises\PhoneBook 目录复制到 C:\SimplyJava 目录中。
- 打开 phoneBook.jsp 模板文件 在文本编辑器中打开 phoneBook.jsp 文件。
- 设置 phoneBook.jsp 的标题 添加 title 元素, 将 JSP 标题设置为 "Phone Book"。
- 为 phoneBook.jsp 添加 header 元素 h1 添加一个显示内容为 "Phone Book Web Application" 的 header 元素 h1。
- 为 phoneBook.jsp 添加表单 添加一个 HTML form 元素。
- 为 phoneBook.jsp 添加文本段 在 form 元素的内部, 添加一个用于显示 "Select a name from the list and click the Get Number button" 的文本段。
- 为 phoneBook.jsp 添加菜单控件 在 form 元素的内部, 添加一个 select 元素, 将其 name 属性设置为 "personName"。
- 为 phoneBook.jsp 添加 submit 按钮 在 form 元素的内部, 添加一个 "submit" 按钮。按钮上所显示的文本为 "Get Number"。"submit" 按钮应出现在菜单控件之后的一个文本段内。
- 保存文件 保存修改后的 phoneBook.jsp 文件。
- 打开 phoneNumber.jsp 模板文件 在文本编辑器中打开 phoneNumber.jsp 文件。
- 设置 phoneNumber.jsp 的标题 添加 title 元素, 将 JSP 的标题设置为 "Phone Number"。
- 为 phoneNumber.jsp 添加一个 header 元素 h1 添加显示内容为 "Phone Number:" 的 header 元素 h1。

- m) 为 `phoneNumber.jsp` 添加文本段 添加一个显示 "numbers" 的文本段。
 - n) 为 `phoneNumber.jsp` 添加超链接 添加一个链接到 "phoneBook.jsp" 的超链接(通过文本 "Phone Book" 实现)。此超链接应位于出现 "numbers" 文本段的某个文本段内。
 - o) 保存文件 保存修改后的 `phoneNumber.jsp` 文件。
 - p) 将电话号码簿应用程序复制到 Tomcat 的 `webapps` 目录中 为安装电话号码簿应用程序, 需要将目录 `C:\SimplyJava\PhoneBook` 复制到 Tomcat 的 `webapps` 目录中。`webapps` 目录位于 Tomcat 默认安装目录 `C:\Program Files\Apache Group\Tomcat 4.1\webapps` 内。
 - q) 启动 Tomcat 选取 `Start → Programs → Apache Tomcat 4.1 → Start Tomcat`。
 - r) 测试应用程序 打开 Web 浏览器, 在 URL 地址栏中分别输入 `http://localhost:8080/PhoneBook/phoneBook.jsp` 和 `http://localhost:8080/PhoneBook/phoneNumber.jsp` 完成对应用程序的测试。
 - s) 停止 Tomcat 选取 `Start → Programs → Apache Tomcat 4.1 → Stop Tomcat` 停止 Tomcat 服务器。
- 30.12 (美国各州知识应用程序: GUI) 创建美国各州知识应用程序的用户界面。有关此应用程序的详细介绍请参见习题 29.12。为该应用程序设计的两个页面如图 30.24 所示。我们将在随后的两个教程中进一步完善此应用程序, 所以, 此时尚不能从数据库中读取州名、图片及其他信息。为帮助读者更好地理解本应用程序, 图 30.25 给出了最终完成设计以后的屏幕效果图。
- a) 将模板复制到工作目录中 将 `C:\Examples\Tutorial30\Exercises\StateFacts` 目录复制到 `C:\SimplyJava` 目录中。
 - b) 打开 `states.jsp` 模板文件 在文本编辑器中打开 `states.jsp` 文件。
 - c) 设置 `states.jsp` 的标题 添加 `title` 元素, 将 JSP 标题设置为 "State List"。
 - d) 为 `states.jsp` 添加 header 元素 `h1` 添加显示内容为 "States" 的 header 元素 `h1`。
 - e) 为 `states.jsp` 添加表单 添加一个 `form` 元素。
 - f) 为 `states.jsp` 添加文本段 在 `form` 元素的内部, 添加一个显示内容为 "Select a state from the list and click the button to view facts about that state" 的文本段。
 - g) 为 `states.jsp` 添加菜单控件 在 `form` 元素的内部, 添加一个 `select` 元素, 其 `name` 属性为 "stateName"。

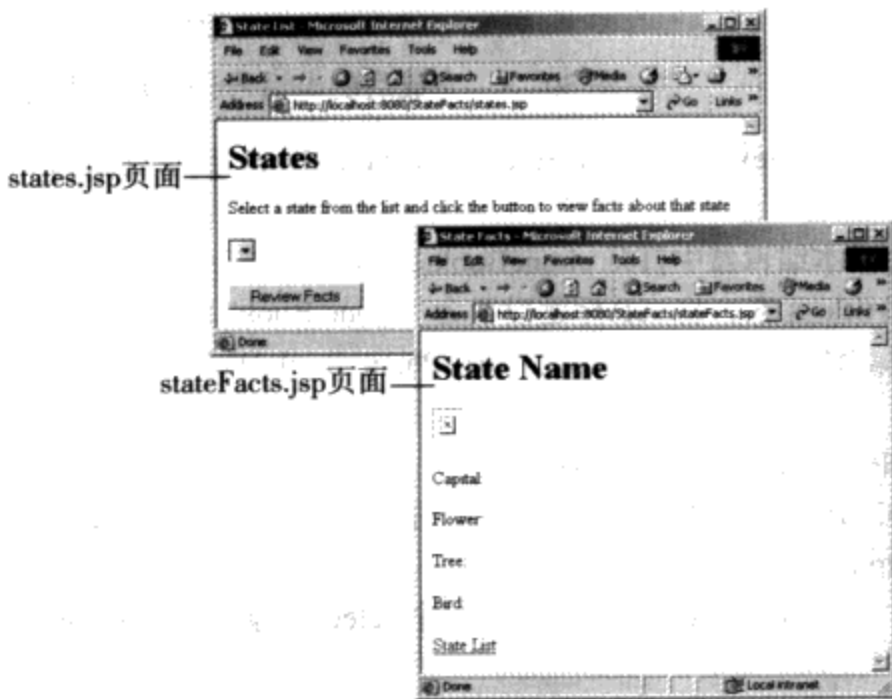


图 30.24 美国各州知识应用程序的用户界面

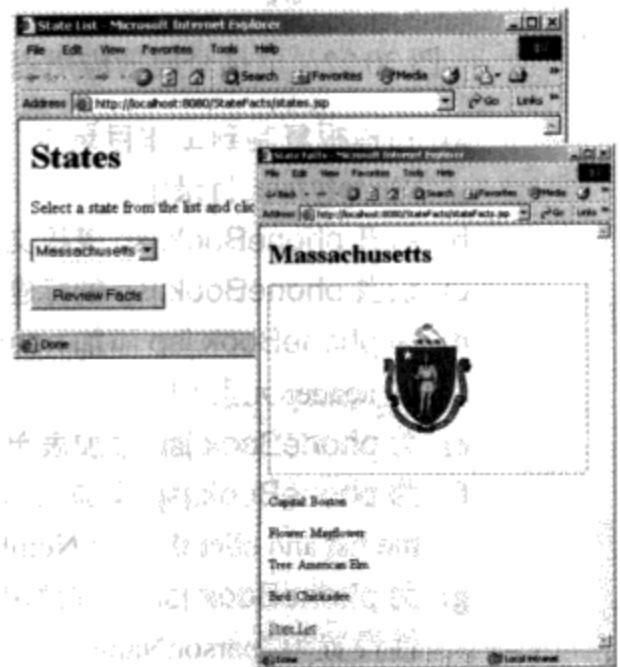


图 30.25 完成后的美国各州知识应用程序

- h) 为 `states.jsp` 添加 submit 按钮 在 `form` 元素的内部, 添加一个 "submit" 按钮。按钮上所显示的文本为 "Review Facts"。
- i) 保存文件 保存修改后的 `states.jsp` 文件。
- j) 打开 `stateFacts.jsp` 模板文件 在文本编辑器中打开 `stateFacts.jsp` 文件。
- k) 设置 `stateFacts.jsp` 的标题 添加 `title` 元素, 将 JSP 的标题设置为 "State Facts"。

- l) 为 stateFacts.jsp 添加 header 元素 h1 添加显示内容为 "State Name" 的 header 元素 h1。
 - m) 为 stateFacts.jsp 添加图片 添加 img 元素，为 JSP 添加一幅图片。此时，src 和 alt 属性的值为 "" (空字符串)，在随后的两个教程中将对它们进行修改。
 - n) 为 stateFacts.jsp 添加 4 个文本段 分别添加 4 个用于显示 "Capital:", "Flower:", "Tree:", "Bird:" 的文本段。
 - o) 为 stateFacts.jsp 添加超链接 添加一个链接至 "states.jsp" 的超链接 (通过文本 "State List" 实现)。
 - p) 保存文件 保存修改后的 stateFacts.jsp 文件。
 - q) 将美国各州知识应用程序复制到 Tomcat 的 webapps 目录中 为安装美国各州知识应用程序，需要将目录 C:\Simplyjava\StateFacts 复制到 Tomcat 的 webapps 目录中。此 webapps 目录位于 Tomcat 默认安装目录 C:\Program Files\Apache Group\Tomcat 4.1\webapps 内。
 - r) 启动 Tomcat 选取 Start → Programs → Apache Tomcat 4.1 → Start Tomcat。
 - s) 测试应用程序 打开一个 Web 浏览器，在 URL 地址栏中分别输入 http://localhost:8080/StateFacts/states.jsp 和 http://localhost:8080/StateFacts/stateFacts.jsp 完成对应用程序的测试。
 - t) 停止 Tomcat 选取 Start → Programs → Apache Tomcat 4.1 → Stop Tomcat 停止 Tomcat 服务器。
- 30.13 (道路标志预览应用程序: GUI) 创建道路标志预览应用程序的用户界面。有关此应用程序的详细介绍请参考练习 29.13。为该应用程序设计的两个页面，如图 30.26 所示。我们将在随后的两个教程中进一步完善此应用程序，所以，此时尚不能从数据库中读取图片。为帮助读者更好地理解本应用程序，图 30.27 给出了最终完成设计以后的屏幕效果图。

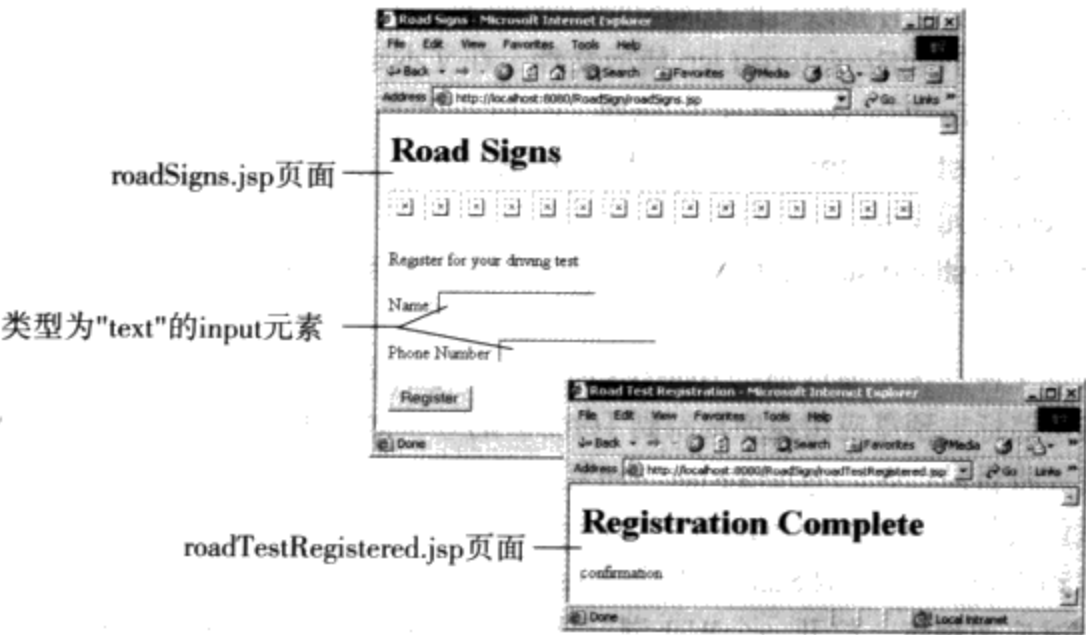


图 30.26 道路标志预览应用程序的用户界面

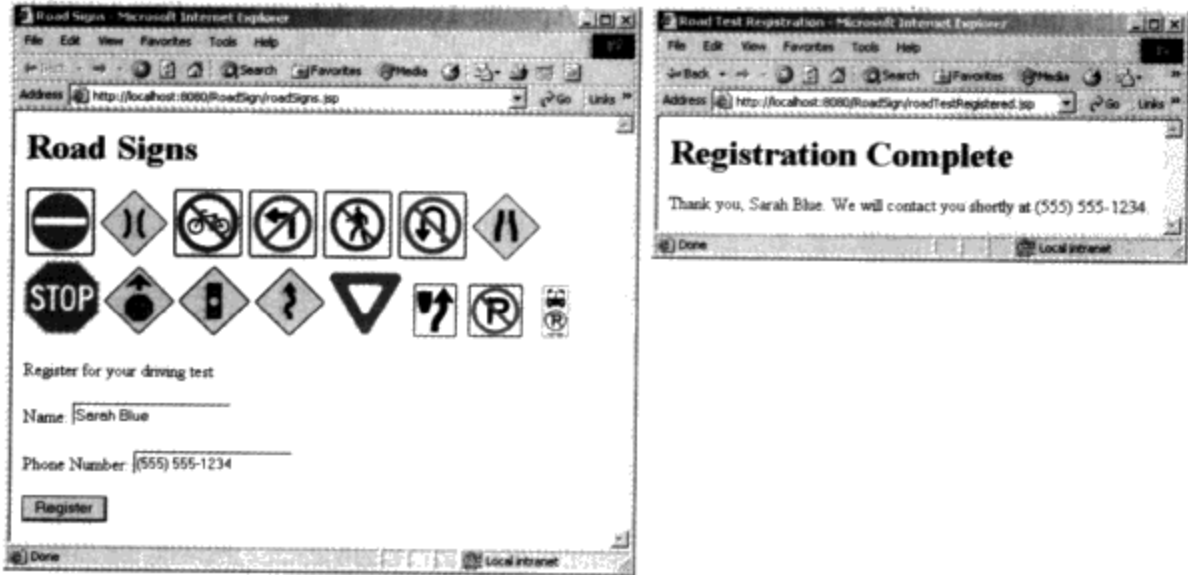


图 30.27 完成后的道路标志预览应用程序

- a) 将模板复制到工作目录中 将 C:\Examples\Tutorial30\Exercises\RoadSign 目录复制到 C:\SimplyJava 目录中。
- b) 打开 roadSigns.jsp 模板文件 在文本编辑器中打开 roadSigns.jsp 文件。
- c) 设置 roadSigns.jsp 的标题 添加 title 元素, 将 JSP 的标题设置为 "Road Signs"。
- d) 为 roadSigns.jsp 添加 header 元素 h1 添加一个显示内容为 "Road Signs" 的 header 元素 h1。
- e) 为 roadSigns.jsp 添加道路标志图片 添加 15 个 img 元素, 将道路标志图片加入到 JSP 中。设置每一个元素的 src 和 alt 属性为 ""。
- f) 为 roadSigns.jsp 添加表单 添加一个 form 元素。
- g) 为 roadSigns.jsp 添加文本段 在 form 元素的内部, 添加一个显示内容为 "Register for your driving test." 的文本段。
- h) 为 roadSigns.jsp 添加文本段和 input 元素 在 form 元素的内部, 添加代码 `<p>Name:<input type="text" name="name"></p>`, 加入一个类型为 "text" 的 input 元素, 显示一个用于输入用户名的文本输入框。
- i) 为 roadSigns.jsp 再次添加文本段和 input 元素 在 form 元素的内部, 通过添加代码 `<p>Phone Number: <input type="text" name="phoneNumber"></p>`, 加入一个类型为 "text" 的 input 元素, 显示一个用于输入用户电话号码的文本输入框。
- j) 为 roadSigns.jsp 添加 submit 按钮 在 form 元素的内部, 添加一个 "submit" 按钮。按钮上所显示的文本为 "Register"。
- k) 保存文件 保存修改后的 roadSigns.jsp 文件。
- l) 打开 roadTestRegistered.jsp 模板文件 在文本编辑器中打开 roadTestRegistered.jsp 文件。
- m) 设置 roadTestRegistered.jsp 的标题 添加 title 元素, 将 JSP 的标题设置为 "Road Test Registration"。
- n) 为 roadTestRegistered.jsp 添加 header 元素 h1 添加一个显示内容为 "Registration Complete" 的 header 元素 h1。
- o) 为 roadTestRegistered.jsp 添加文本段 在 form 元素的内部, 添加一个显示内容为 "confirmation" 的文本段。
- p) 保存文件 保存修改后的 roadTestRegistered.jsp 文件。
- q) 将道路标志预览应用程序复制到 Tomcat 的 webapps 目录中 为安装道路标志预览应用程序, 需要将目录 C:\SimplyJava\RoadSign 复制到 Tomcat 的 webapps 目录中。此 webapps 目录位于 Tomcat 默认安装目录 C:\Program Files\Apache Group\Tomcat 4.1\webapps 内。
- r) 启动 Tomcat 选取 Start → Programs → Apache Tomcat 4.1 → Start Tomcat。
- s) 测试应用程序 打开一个 Web 浏览器分别在 URL 地址栏中输入 `http://localhost:8080/RoadSign/roadSigns.jsp` 和 `http://localhost:8080/RoadSign/roadTestRegistered.jsp` 完成对应用程序的测试。
- t) 停止 Tomcat 选取 Start → Programs → Apache Tomcat 4.1 → Stop Tomcat 停止 Tomcat 服务器。



教程 31 Web 书店应用程：信息层

考察数据库并创建数据库组件

教学目标

在本教程中，读者将学到以下内容：

- 连接数据库
- 创建查询数据库信息的 SQL 语句

在本教程中将集中讨论 Web 书店应用程序的信息层，该层存储了应用程序中的全部数据。在 Web 书店应用程序中，信息层使用的是 Cloudscape 数据库 bookstore，数据库中存储了每本书的信息。在开始讲述本教程之前，应掌握教程 26 中有关数据库的概念。

在本教程中，需要创建用以连接数据库及执行 SQL 语句的 Connection 对象和 Statement 对象。同时，还将定义查询数据库数据的 SQL 语句。事实上，该信息层仅包括 bookstore 数据库。本教程中所创建的 Connection 对象和 Statement 对象实际是属于中间层的一部分，它们执行从数据库中查询数据的任务。创建这些对象的目的是为了能够同信息层进行交互。在下一教程中，通过创建中间层将完成整个 Web 书店应用程序的开发。

31.1 回顾 Web 书店应用程序

我们曾在教程 29 中探讨过这个三层 Web 书店应用程序，并在教程 30 中利用 HTML 设计出了它的 GUI。下面，将着手完成此应用程序的数据库组件。在本教程中，将创建用于从数据库中获取信息的一些对象，包括连接数据库的对象和执行 SQL 语句的对象。在开始这项工作之前，先回顾一下教程 30 中的 ACE 表及其伪代码，其中列出了完成该应用程序所需的操作、组件及其事件。

31.2 信息层：数据库

信息层中保存了应用程序所需的数据。数据库中储存的数据可包括产品数据（如产品介绍、价格、库存量，等等）和客户数据（如客户姓名、发货信息，等等）。

数据库是实际应用程序中必不可少的一部分。一旦数据被录入数据库，只有具备一定权限的用户和应用程序才可对其进行访问。由于数据以电子格式存储，因此存取或处理的速度要比使用纸介质快得多。大多数的数据库都属于关系型数据库——数据被组织在相互关联的表中。存在许多可用的数据库产品（实现数据库的创建和修改），范围从个人版的 Microsoft Access 到企业级的 Oracle, Sybase, IBM 的 DB2, Microsoft SQL Server, 等等。另外，数据库产品还能够生成数据库信息报告。

Web 书店应用程序中的所有图书数据被储存在一个称为 bookstore 的 Cloudscape 数据库中。使用 Java 代码和 JDBC API（一种用于数据库的 Java 类集）能够检索出数据库中的数据。bookstore 数据库含有一张名为 products 的数据表，表中记录了每本书的详细信息。

products 表共有 9 列: productID, title, authors, copyrightYear, edition, isbn, cover, description, price。这些列分别代表 ID 编号、书名、作者、版权期限、版次、ISBN (惟一标识每种书的编码)、封面图片的文件名、描述信息及价格。products 表的内容如图 31.1 和图 31.2 所示。图 31.1 为 products 表前 3 列的数据。图 31.2 为 products 表剩余 6 列的数据。

productID	title	authors
1	Visual Basic .NET How to Program: Second Edition	Harvey M. Deitel, Paul J. Deitel & Tem R. Nieto
2	C++ How to Program: Fourth Edition	Harvey M. Deitel & Paul J. Deitel
3	C# How to Program: First Edition	Harvey M. Deitel, Paul J. Deitel, Jeff Listfield, Tem R. Nieto, Cheryl Yaeger & Marina Zlatkina

图 31.1 bookstore 数据库中的 products 表

copyright-Year	edition	isbn	cover	description	price
2002	2	0-13-029363-6	vbnethttp2.png	Microsoft Visual Basic .NET	75.99
2002	4	0-13-038474-7	cpphttp4.png	Introduces Web programming with CGI and object-oriented design with the UML.	75.99
2002	1	0-13-062221-4	csharphttp1.png	Introduces .NET and Web services	75.99

图 31.2 bookstore 数据库中的 products 表 (续)

自测题

- 1. 一旦数据录入数据库，_____。
 - a) 拥有适当权限的用户和应用程序可以进行访问
 - b) 用户必须等待系统重新启动才能访问数据
 - c) 使用数据库的应用程序必须手动更新
 - d) 以上答案都不对
- 2. 在关系型数据库中，数据组织在_____中。
 - a) 类 b) 表 c) 结果集 (Result Set) d) 以上答案都不对

答案: 1) a 2) b

31.3 在 JSP 页面中使用 Cloudscape 数据库

在开始编写 Web 书店应用程序的中间层之前，首先必须建立与数据库之间的连接，从而检索数据库中的数据。使用 JDBC 创建连接数据库的 Connection 对象和执行 SQL 语句的 Statement 对象。然后，利用 ResultSet 对象处理 SQL 语句返回的结果。

在 books.jsp 页面中添加数据库组件

- 1. 将模板复制到工作目录中 将 C:\Examples\Tutorial31\TemplateApplication\bookstore_informationTier 目录复制到 C:\SimplyJava 目录中。
- 2. 打开 books.jsp 模板文件 在文本编辑器中打开 books.jsp 模板文件。
- 3. 添加 JSP 脚本 将图 31.3 中第 33 行至第 44 行添加到 JSP 中。此段代码添加了一段 JSP 脚本，实现数据库连接与处理的操作。利用脚本程序员可以在 JSP 中插入 Java 代码。
第 30 行为 JSP 注释。JSP 注释使用<%-- 和 --%>表示，整个 JSP 中都可以放置 JSP 注释和 HTML 注释，但不能放在 JSP 脚本内。脚本是由位于<% 和 %>之间的 Java 代码所组成的。第 31 行至第 44 行创建了一段 JSP 脚本。脚本内可包含 // 注释或 /*, */ 注释。第 33 行至第 36 行定义的 try 语句块，其内将创

建 Connection, Statement 和 Resultset 对象。第 39 行至第 42 行的 catch 语句块捕获从 try 块中抛出的任何 SQLException 异常。



图 31.3 针对 JSP 脚本添加模板

为使用 SQLException 及其他与数据库连接和处理有关的类，需要导入 java.sql 包。模板中已经提供了导入 java.sql 包的代码。此代码位于图 31.21 中第 5 行。

4. 指定数据库位置 将图 31.4 中第 35 行至第 37 行添加到 JSP 中。Cloudscape JDBC 驱动程序根据系统属性 db2j.system.home，确定数据库文件的位置。第 36 行至第 37 行调用 System 的 setProperty 方法将该属性设置为 bookstore 数据库所在的位置。方法 setProperty 接收两个 String 型变量——一个表示系统属性的名称，另一个指出设定的值。在本例中，数据库位于 C:\Examples\Tutorial29\Databases 目录内。第 37 行，注意字符串中所使用的转移字符\\，表示插入一个字符\ [注意：如果教程 29 的位置并非 C:\Examples，那么请将 C:\\Examples 目录（图 31.4 中第 37 行）替换为教程 29 所在的目录]。

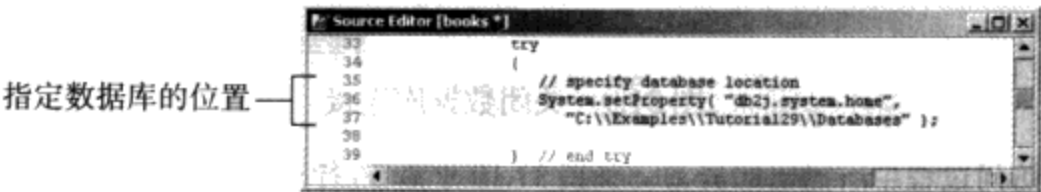


图 31.4 设置数据库的位置

5. 载入 Cloudscape 数据库驱动程序 将图 31.5 中第 39 行至第 40 行添加到 JSP 页面中。第 40 行使用 Class 类的 forName 方法载入 Cloudscape JDBC 驱动程序。

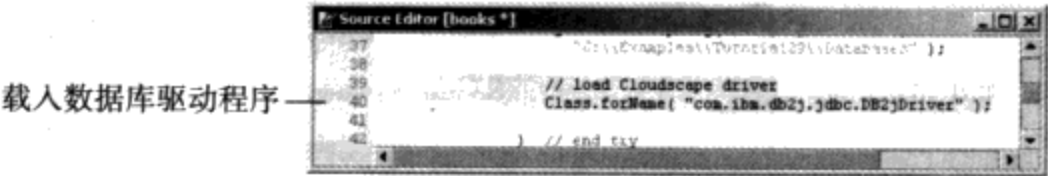


图 31.5 载入数据库驱动程序

6. 创建 Connection 对象 在访问数据库之前必须先建立同数据库之间的连接。读者将使用 JDBC Connection 对象完成此项任务。Connection 对象可创建一个用于从数据库中获取图书信息的 Statement 对象。将图 31.6 中第 42 行至第 45 行添加到 JSP 中，创建连接 bookstore 数据库的 Connection 对象。以前曾经讲过，传递给 getConnection 方法的参数是一个代表数据库名和数据库访问协议的 JDBC URL。
7. 创建 Statement 对象 将图 31.7 中第 47 行至第 53 行添加到 JSP 页面中。第 48 行查看上一步中的连接操作是否成功。第 50 行至第 51 行创建了一个 Statement 对象执行 SQL 语句。

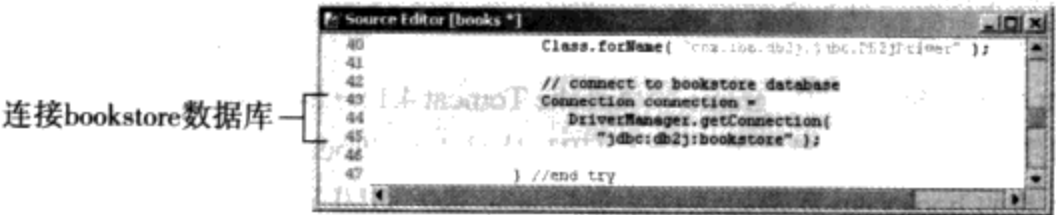


图 31.6 连接数据库

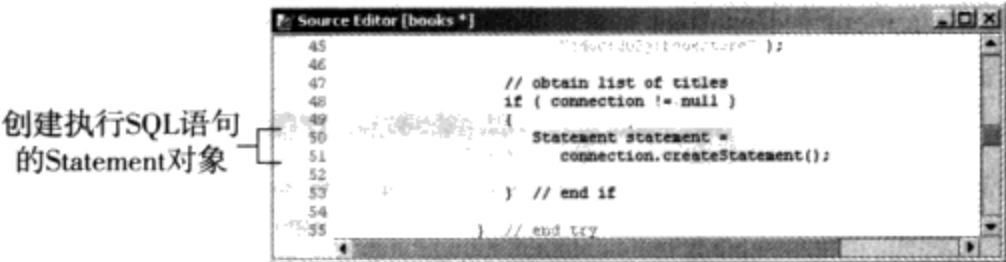


图 31.7 创建 Statement 对象

8. 执行查询 利用 Statement 对象的 executeQuery 方法执行从数据库中获取信息的查询。executeQuery 方法返回一个含查询结果的 ResultSet 对象。添加图 31.8 中第 53 行至第 54 行，从 products 表中查找书名。在教程 32 中，将在 books.jsp 页面中添加处理 ResultSet 的代码，以完成相应的菜单控件。

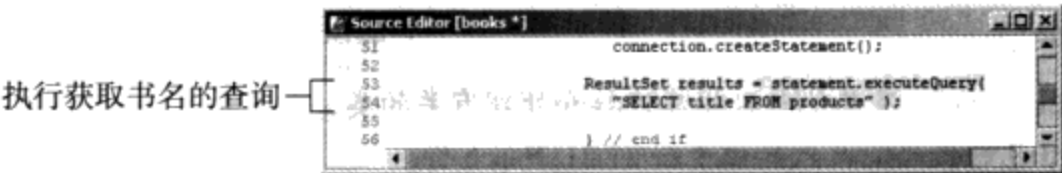


图 31.8 获取数据库中的信息

9. 关闭数据库连接 处理完数据以后，关闭数据库连接释放相应的数据库资源。将图 31.9 中第 58 行添加到 JSP 页面中。此行代码调用 Connection 类的 close 方法关闭数据库连接。

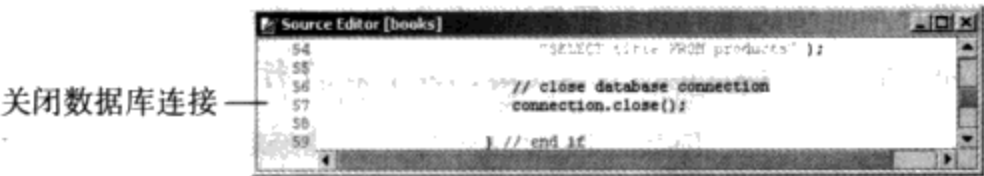


图 31.9 关闭数据库连接

10. 捕获数据库访问过程中所抛出的异常 将图 31.10 中第 66 行至第 67 行添加到 JSP 页面中。在 try 语句块中，如果有异常抛出，第 66 行至第 67 行的代码会显示一条与之相应的错误信息。第 66 行的 out 对象是 JSP 中的一个隐含对象。隐含对象是一套供程序员使用的 Java 对象，可完成 JSP 页面中对客户端的请求和响应。程序员无需直接创建就可以使用隐含对象。out 对象利用 println 方法将写入的字符串作为响应请求的一部分。

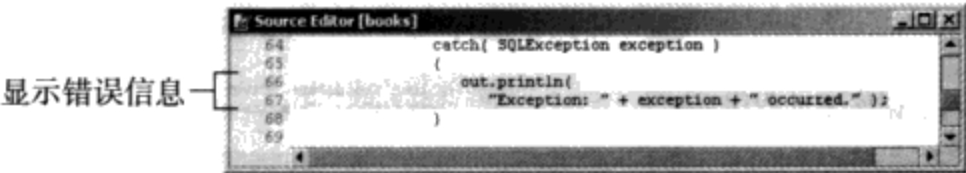


图 31.10 JSP 脚本的结束及异常的捕获

- 11. 保存应用程序 保存修改后的源代码文件。
- 12. 将 books.jsp 复制到 bookstore 目录中 将修改后的 books.jsp 文件复制到 C:\Program Files\Apache Group\Tomcat 4.1\webapps\bookstore 目录下。
- 13. 将数据库 JAR 文件复制到 Web 书店应用程序中 Web 书店应用程序的信息层使用的是 Cloudscape 数据库。需要将 db2j.jar 和 license.jar 文件复制到 C:\Program Files\Apache Group\Tomcat 4.1\webapps\bookstore\WEB-INF\lib 目录下。db2j.jar 和 license.jar 文件位于 C:\Cloudscape_5.1\lib 目录下。在教程 26 中，如果未将 Cloudscape 安装在 C:\Cloudscape_5.1 目录下，需将 C:\Cloudscape_5.1 目录替换为 Cloudscape 的安装目录。
- 14. 启动 Tomcat 选取 Start → Programs → Apache Tomcat 4.1 → Start Tomcat 启动 Tomcat 服务器。
- 15. 测试应用程序 打开 Web 浏览器，输入 URL 地址 http://localhost:8080/bookstore/books.jsp。图 31.11 显示了更新后的 books.jsp 页面的运行结果。注意，客户 GUI 同以前一样，因为此时尚未将书名添加到菜单控件中。我们将在下一教程中完善此项功能。

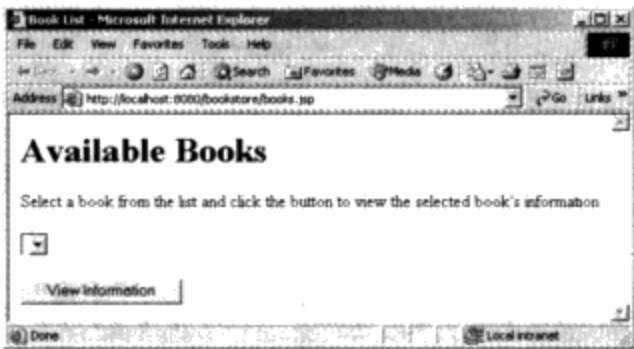


图 31.11 运行更新后的 books.jsp 页面

- 16. 关闭浏览器 点击浏览器窗口的关闭按钮关闭浏览器。
- 17. 停止 Tomcat 选取 Start → Programs → Apache Tomcat 4.1 → Stop Tomcat 停止 Tomcat 服务器。

到目前为止，我们已在 books.jsp 页面中添加了与数据库处理有关的 Connection 对象和 Statement 对象，下面，将在 bookInformation.jsp 页面中完成同样的功能。

在 bookInformation.jsp 页面中添加数据库组件

- 1. 打开 bookInformation.jsp 模板文件 在文本编辑器中打开 bookInformation.jsp 模板文件（位于目录 C:\SimplyJava\bookstore_informationTier 中）。
- 2. 起始 JSP 脚本 将图 31.12 中第 23 行至第 27 行添加到 JSP 页面中。此代码段将开始一段连接数据库的 JSP 脚本。

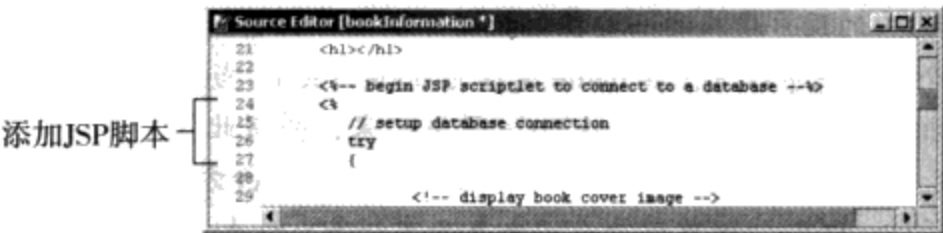


图 31.12 针对 JSP 脚本添加模板

- 3. 指定数据库的位置 将图 31.13 中第 28 行至第 30 行添加到 JSP 页面中。第 29 行至第 30 行指定数据库的位置 [注意：如果教程 29 的位置并非 C:\Examples，那么请将 C:\Examples 目录（图 31.4 中第 37 行）替换为教程 29 所在的目录]。

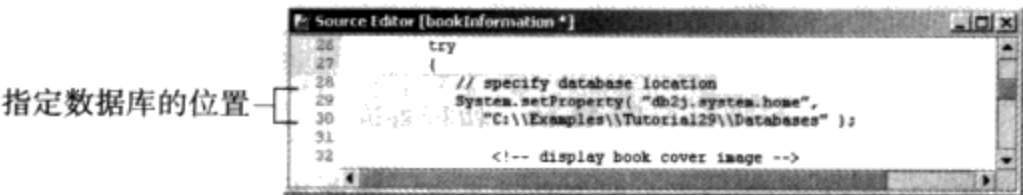


图 31.13 指定数据库的位置

- 4. 载入 Cloudscape 驱动程序 将图 31.14 中第 32 行至第 33 行添加到 JSP 页面中。第 33 行载入 Cloudscape JDBC 驱动程序。

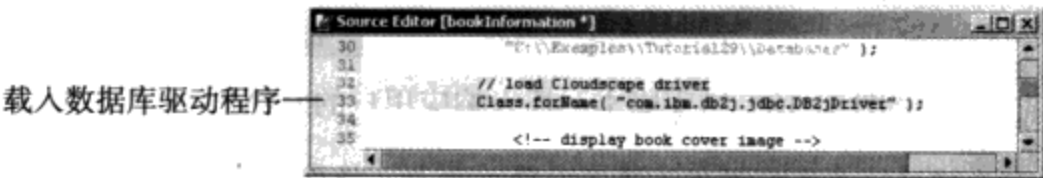


图 31.14 载入数据库驱动程序

- 5. 创建 Connection 对象 将图 31.15 中第 35 行至第 37 行添加到 JSP 页面中。在访问数据之前，必须与数据库建立连接。此功能需使用 Connection 对象来完成。利用 Connection 从数据库中读取图书信息。第 35 行至第 37 行调用 DriverManager 类的 getConnection 方法建立同 bookstore 数据库的连接。

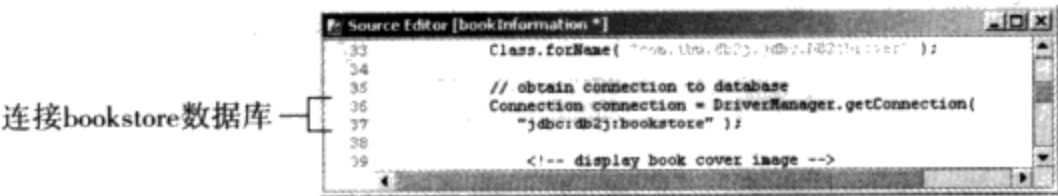


图 31.15 建立 JDBC 连接

6. 创建 Statement 对象 将图 31.16 中第 39 行至第 43 行添加到 JSP 页面中。第 40 行判断 Connection 对象是否创建成功。如果创建成功，则第 43 行会得到一个 Statement 对象。

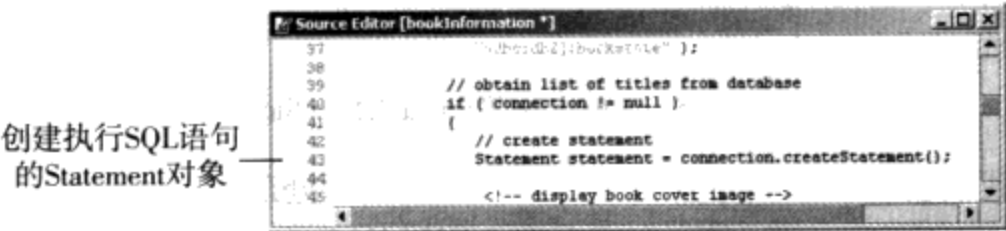


图 31.16 创建 Statement 对象

7. 执行查询 下面，执行从数据库中获取所选图书信息的查询。将图 31.17 中第 45 行至第 50 行添加到 JSP 页面中。该语句针对用户在 books.jsp 页面中所选的书名，从 products 表中检索 cover, title, authors, price, isbn, edition, copyrightYear, description 列中的信息。当指定 SQL 语句时，第 50 行使用 JSP 隐含对象 request 的 getParameter 方法取得参数 bookTitle 的值。方法 getParameter 接收一个 String 型参数，并以字符串形式返回此参数的值。隐含对象 request 表示来自 Web 浏览器客户端的请求，并能够访问客户端 Web 页面中的用户输入和一些其他数据。我们曾在教程 30 中的图 30.6 内，使用 HTML select 元素创建了一个空白菜单控件（其 name 属性为 bookTitle）。当用户在 books.jsp 中点击 "submit" 按钮时，浏览器将把用户所选书名同 bookTitle 联系起来，并将此信息作为请求的一部分传递给 book-Information.jsp。调用参数为 "bookTitle" 的 request.getParameter 方法，可返回用户在 books.jsp 中所选书的书名。曾经讲过，SQL 使用单引号 (') 来表示字符串。因此，第 49 行至第 50 行使用单引号表示由 getParameter 方法返回的字符串。

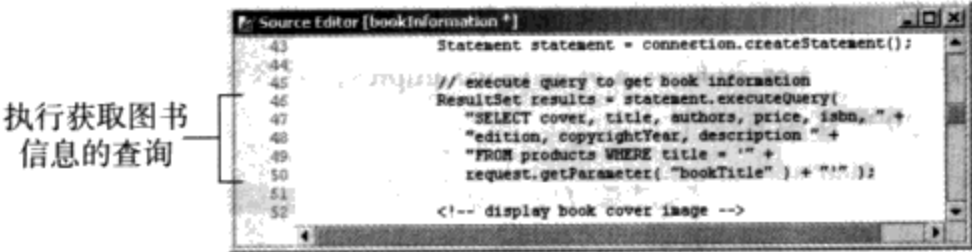


图 31.17 执行查询

8. 结束脚本 将图 31.18 中第 52 行添加到 JSP 页面中，结束此段脚本。结束脚本是因为 HTML 标记不能放置在 JSP 脚本中。JSP 脚本只能包含 Java 代码。在下一步中，将添加关闭数据库连接的脚本。

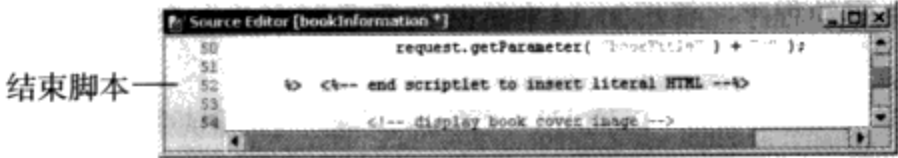


图 31.18 结束脚本，插入响应用户的 HTML 标记

9. 关闭连接 前面讲过，关闭数据库连接可以释放资源。将图 31.19 中第 78 行至第 92 行添加到 JSP 页面中。第 78 行至第 92 行将继续前面的脚本（参见第 78 行），提供关闭数据库连接（参见第 80 行）和异常处理的操作（参见第 87 行至第 90 行）。第 80 行调用 close 方法关闭 Connection 对象。第 82 行结束起始于第 40 行的 if 语句（如图 31.16 所示）。第 84 行结束起始于第 26 行的 try 语句（如图 31.12 所示）。第 87 行至第 90 行捕获 try 语句中抛出的异常。

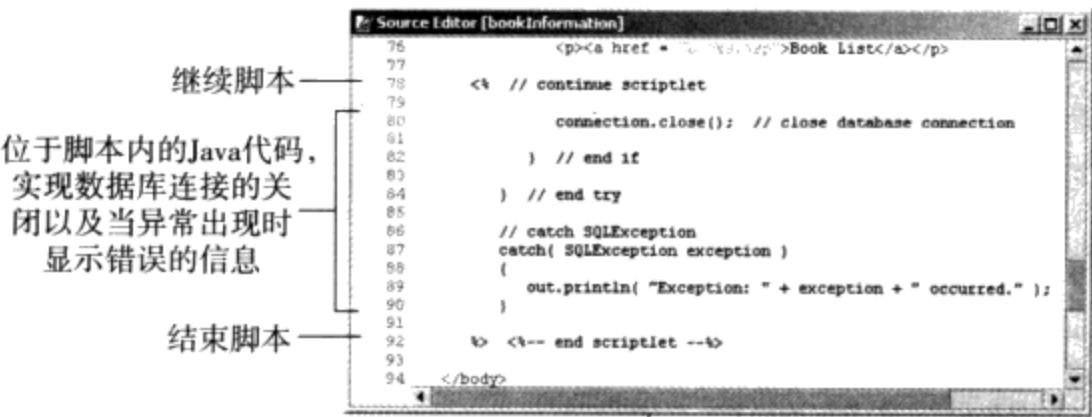


图 31.19 关闭数据库连接

- 10. 保存应用程序 保存修改后的源代码文件。
- 11. 将 bookInformation.jsp 复制到 bookstore 目录中 将修改后的 bookInformation.jsp 文件复制到 C:\ProgramFiles\Apache Group\Tomcat 4.1\webapps\bookstore 目录中。
- 12. 将数据库 JAR 文件复制到 Web 书店应用程序中 Web 书店应用程序的信息层使用的是 Cloudscape 数据库。需要将 db2j.jar 和 license.jar 文件复制到 C:\Program Files\Apache Group\Tomcat4.1\webapps\bookstore\WEB-INF\lib 目录下。db2j.jar 和 license.jar 文件位于 C:\Cloudscape_5.1\lib 目录下。在教程 26 中, 如果未将 Cloudscape 安装在 C:\Cloudscape_5.1 目录下, 需将目录 C:\Cloudscape_5.1 替换为 Cloudscape 的安装目录。
- 13. 启动 Tomcat 选取 Start → Programs → Apache Tomcat 4.1 → Start Tomcat 启动 Tomcat 服务器。
- 14. 测试应用程序 打开一个 Web 浏览器, 输入 URL 地址 http://localhost:8080/bookstore/bookInformation.jsp。图 31.20 显示了更新后的 bookInformation.jsp 页面的运行结果。注意, 其结果同教程 30 中的结果一样, 这是因为尚未插入响应客户的图书信息。



图 31.20 更新后的 bookInformation.jsp 页面的运行结果

- 15. 关闭浏览器 点击浏览器窗口的关闭按钮关闭浏览器。
- 16. 停止 Tomcat 选取 Start → Programs → Apache Tomcat 4.1 → Stop Tomcat 停止 Tomcat 服务器。

图 31.21 和图 31.22 中分别显示了 books.jsp 和 bookInformation.jsp 文件的源代码。在本教程中, 凡需要添加、查看或者修改的代码, 均在图中进行了突出显示。

```
1 <!-- Tutorial 31: books.jsp -->
2 <!-- Displays a form. -->
3
4 <%-- import java.sql.* for database classes --%>
5 <%@ page import = "java.sql.*" %>
6
7 <!-- begin HTML document -->
8 <html>
```

导入包 java.sql 中的类


```

68         }
69
70         %> <!-- end scriptlet --%>                                结束脚本
71
72         </select>
73
74         <!-- create View Information button -->
75         <p><input type = "submit" value = "View Information"></p>
76     </form>
77
78 </body>
79 </html>

```

图 31.21 books.jsp 源代码

```

1 <!-- Tutorial 31: bookInformation.jsp -->
2 <!-- Displays book information. -->
3
4 <!-- import java.sql.* for database classes --%>
5 <%@ page import = "java.sql.*" %>                                导入包 java.sql 中的类
6
7 <!-- begin HTML document -->
8 <html>
9
10 <!-- specify head element -->
11 <head>
12
13 <!-- specify page title -->
14 <title>Book Information</title>
15 </head>
16
17 <!-- begin body of document -->
18 <body>
19
20 <!-- create a heading for the book's title -->
21 <h1></h1>
22
23 <!-- begin JSP scriptlet to connect to a database --%>          JSP注释
24 <%                                                                起始脚本
25     // setup database connection
26     try
27     {
28         // specify database location
29         System.setProperty( "db2j.system.home",
30             "C:\\Examples\\Tutorial29\\Databases" );                指定数据库位置
31
32         // load Cloudscape driver
33         Class.forName( "com.ibm.db2j.jdbc.DB2jDriver" );          载入数据库驱动程序
34
35         // obtain connection to database
36         Connection connection = DriverManager.getConnection(
37             "jdbc:db2j:bookstore" );                                连接 bookstore 数据库
38
39         // obtain list of titles from database
40         if ( connection != null )
41         {
42             // create statement
43             Statement statement = connection.createStatement();    创建执行 SQL 查询的 Statement 对象
44

```

```

45      // execute query to get book information
46      ResultSet results = statement.executeQuery(
47          "SELECT cover, title, authors, price, " +
48          "isbn, edition, copyrightYear, description " +
49          "FROM products WHERE title = '" +
50          request.getParameter( "bookTitle" ) + "'";
51
52      %> <%-- end scriptlet to insert literal HTML --%>
53
54          <!-- display book cover image -->
55          <img src = "" alt = "">
56
57          <!-- display authors -->
58          <p>Author(s): </p>
59
60          <!-- display price -->
61          <p>Price: </p>
62
63          <!-- display ISBN -->
64          <p>ISBN: </p>
65
66          <!-- display edition number -->
67          <p>Edition: </p>
68
69          <!-- display copyright year -->
70          <p>Copyright Year: </p>
71
72          <!-- display authors -->
73          <p>Description: </p>
74
75          <!-- create link to Book List -->
76          <p><a href = "books.jsp"> Book List</a></p>
77
78      <% // continue scriptlet
79
80          connection.close(); // close database connection
81
82      } // end if
83
84      } // end try
85
86      // catch SQLException
87      catch ( SQLException exception )
88      {
89          out.println( "Exception: " + exception + " occurred." );
90      }
91
92      %> <%-- end scriptlet --%>
93
94      </body>
95 </html>

```

执行获取
图书信息
的查询

结束脚本开始插入 HTML 标记

继续脚本

位于脚本内的 Java 代码，
实现数据库连接的关闭以及
当异常出现时显示错误信息

结束脚本

图 31.22 bookInformation.jsp 源代码

自测题

- _____ 注释不能在脚本中使用。
a) HTML b) Java c) JSP d) (a)和(c)
- JSP 脚本以 _____ 进行界定。

- a) <%-- 和 --%> b) <% 和 %> c) <!-- 和 --> d) 以上答案都不对

答案：1) d 2) b

31.4 小结

在本教程中，学习了三层 Web 书店应用程序中的信息层，查看了 bookstore 数据库中的内容。通过创建 JDBC Connection 对象和 Statement 对象，访问了 Web 书店应用程序中的信息层。

在下一教程中，将完成 Web 书店应用程序的中间层，通过增添代码，确定 JSP 页面上所显示的数据。

技术小结

添加 JSP 注释

- 将注释文字放置于<%-- 和 --%>之间。

添加 JSP 脚本

- 将脚本放置于<% 和 %>之间。
- 在脚本内部，添加执行任务（如连接数据库）的 Java 代码。

显示错误信息

- 使用隐含对象 out 的 println 方法显示错误信息。

在 JSP 中获取用户输入

- 使用 JSP 隐含对象 request 的 getParameter 方法获取用户的输入。

关键术语

JSP 隐含对象 request 的 getParameter 方法 获取请求参数的值。

隐含对象 JSP 页面中提供程序员访问来自客户端请求及实现对客户端响应的对象。程序员无需显式创建就可以使用的隐含对象。

JSP 注释 位于<%-- 和 --%>之间的文本。

JSP 隐含对象 out 用于将写入的文本作为对用户请求的响应。

JSP 隐含对象 out 的 println 方法 用于将写入的文本作为对用户请求部分的响应。

JSP 隐含对象 request 表示客户端请求。用于获取提交至 JSP 客户端请求中的数据。

脚本 位于<% 和 %>之间的代码块，允许程序员将 Java 代码插入到 JSP 页面中。

System 类的 setProperty 方法 设置系统的属性，如 db2j.system.home。

JSP 参考索引

JSP 注释 使用<%-- 和 --%>进行标注的一种注释。

out 此 JSP 隐含对象用于将写入的文本作为对客户端请求的响应。

- 方法

println 将写入的文本作为对客户端请求的响应。

request 此 JSP 隐含对象表示客户端请求。用于获取提交至 JSP 客户端请求中的数据。

- 方法

getParameter 获取请求参数的值。

脚本 利用 JSP 脚本程序员可在 JSP 中插入一段 Java 代码。脚本需放置在<% 和 %>之间。

习题

选择题

- 31.1 _____ 是一个数据库产品。
a) Cloudscape b) Microsoft SQL Server
c) Oracle d) 以上答案都正确
- 31.2 JSP 中的 _____ 对象为程序员提供了访问客户端请求及客户端响应的功能。
a) 脚本 b) 显式 c) 隐含 d) 以上答案都不对
- 31.3 当注释出现在 JSP 脚本中时, 它一定是 _____。
a) // 注释或 /*, */ 注释 b) HTML 注释
c) JSP 注释 d) 以上答案都正确
- 31.4 系统属性 _____ 用于指定数据库的位置。
a) db2j.system.home b) db2j.system.database.home
c) db2j.system.location d) db2j.system.database.location
- 31.5 _____ 类的 getConnection 方法能够创建一个针对某数据库的 Connection。
a) DatabaseDriver b) CloudscapeManager
c) DriverManager d) 以上答案都不对
- 31.6 使用 _____ 对象可以执行查询数据库中数据的 SQL 语句。
a) Connection b) Statement c) DriverManager d) 以上答案都不对
- 31.7 _____ 用来向 JSP 页面中添加 Java 代码。
a) JSP 注释 b) JSP 脚本 c) (a)和(b) d) 以上答案都不对
- 31.8 由 Statement 类的 executeQuery 方法所返回的 _____ 对象能够查询数据库中的数据。
a) Connection b) Statement c) ResultSet d) 以上答案都不对
- 31.9 隐含对象 request 的 _____ 方法可取得提交给 JSP 作为客户端请求中的数据。
a) getParameter b) getValue c) getField d) getName
- 31.10 数据库层还可被称为 _____。
a) 信息层 b) 底层 c) (a)和(b) d) 以上答案都不对

练习题

- 31.11 (电话号码簿应用程序: 数据库) 利用 JSP 脚本创建电话号码簿应用程序数据库中的 Connection 对象和 Statement 对象。数据库 phonebook 中有一张 phoneNumbers 表, 内含 3 列: id, name 和 phoneNumber。其中, id 号包含惟一标识人员的数字, name 中的字符串表示人员的姓名, phoneNumber 中的字符串则表示电话号码。
- a) 打开 phoneBook.jsp 打开练习 30.11 中创建的 phoneBook.jsp 文件。
- b) 从 java.sql 中导入供 JSP 使用的类 将图 31.21 中第 4 行至第 5 行插入 phoneBooks.jsp 中第 4 行。
- c) 添加 JSP 脚本 在 phoneBook.jsp 源代码中 HTML select 元素的内部添加一段 JSP 脚本, 内含一个 try 语句块和一个捕获从 try 语句块中抛出 SQLException 异常的 catch 语句块。
- d) 添加一个针对数据库的 Connection 在上一步中定义的 try 语句块的内部, 利用值 "C:\Examples\Tutorial31\Exercises\Databases" 指定数据库的位置, 载入数据库驱动程序类并连接 phonebook 数据库 (jdbc:db2j:phonebook) (注意: 如果教程 31 的位置并非 C:\Examples, 那么请使用教程 31 所在的目录替换 C:\Examples)。
- e) 创建 Statement 执行从数据库中检索姓名的查询 在 try 语句块的内部位于数据库连接成功代码的后面, 创建一个执行 SQL 查询的 Statement 对象。执行从数据库中获取所有姓名的查询。
- f) 关闭数据库连接 执行完查询以后, 调用 Connection 对象的 close 方法断开同数据库的连接。
- g) 显示 SQLException 错误消息 在 phoneBook.jsp 中 catch 语句块的内部, 使用 JSP 隐含对象 out 的 println 方法显示 SQLException 异常的错误消息。

- h) **保存文件** 保存修改后的 phoneBook.jsp 文件。
 - i) **打开 phoneNumber.jsp** 打开练习 30.11 中创建的 phoneNumber.jsp 文件。
 - j) **从 java.sql 中导入要使用的类** 将图 31.21 中第 4 行至第 5 行插入 phoneNumber.jsp 中的第 4 行。
 - k) **起始 JSP 脚本** 在 phoneNumber.jsp 源代码的内部, 位于 HTML 元素 h1 的后面, 起始一段 JSP 脚本。在该脚本的内部, 定义一个 try 语句块。
 - l) **添加一个针对数据库的 Connection** 在 try 语句块的内部, 利用值 "C:\Examples\Tutorial31\Exercises\Databases" 指定数据库的位置, 载入数据库驱动程序类并连接 phonebook 数据库(注意: 如果教程 31 的位置并非 C:\Examples, 那么请将 C:\Examples 目录替换为教程 31 所在的目录)。
 - m) **创建 Statement 执行从数据库中检索姓名的查询** 在数据库连接成功代码的后面, 创建一个执行 SQL 查询的 Statement 对象。执行查询, 获取与所选姓名相对应的人的信息。
 - n) **结束脚本** 在 HTML p 元素之前(内容为 "numbers"), 结束起始于步骤(k)的脚本, 这样, 便可放置 HTML 标记并实现对客户端的响应。
 - o) **添加另一个 JSP 脚本** 在 phoneNumber.jsp 源代码的内部, 位于 HTML p 元素中包含超链接至 phoneBook.jsp 代码的后面, 再添加一段脚本, 实现关闭数据库连接、结束 try 语句块以及当 SQLException 异常发生时显示一条错误消息。
 - p) **保存文件** 保存修改后的 phoneNumber.jsp 文件。
 - q) **将 phoneBooks.jsp 和 phoneNumber.jsp 文件复制到 Tomcat webapps 目录下** 将更新后的 phoneBook.jsp 和 phoneNumber.jsp 文件复制到 C:\Program Files\Apache Group\Tomcat4.1\webapps\PhoneBook 目录下。
 - r) **将数据库 JAR 文件复制到电话号码簿 Web 应用程序中** 电话号码簿 Web 应用程序的信息层使用的是 Cloudscape 数据库。需要将 db2j.jar 和 license.jar 文件复制到 C:\Program Files\Apache Group\Tomcat4.1\webapps\PhoneBook\WEB-INF\lib 目录中。db2j.jar 和 license.jar 文件位于 C:\Cloudscape_5.1\lib 目录下。在教程 26 中, 如果 Cloudscape 的安装目录不是 C:\Cloudscape_5.1, 需将 C:\Cloudscape_5.1 目录替换为 Cloudscape 的安装目录。
 - s) **启动 Tomcat** 选取 Start → Programs → Apache Tomcat 4.1 → Start Tomcat 启动 Tomcat 服务器。
 - t) **测试应用程序** 打开一个 Web 浏览器, 分别输入 URL 地址 http://localhost:8080/PhoneBook/phoneBook.jsp 和 http://localhost:8080/PhoneBook/phoneNumber.jsp 完成对应用程序的测试。
 - u) **停止 Tomcat** 选取 Start → Programs → Apache Tomcat 4.1 → Stop Tomcat 停止 Tomcat 服务器。
- 31.12 (美国各州知识应用程序: 数据库) 利用 JSP 脚本创建美国各州知识应用程序数据库中的 Connection 对象和 Statement 对象。数据库 statefacts 中有一张 states 表, 内含 7 列: id, name, flag, capital, flower, tree, bird。其中, id 号中包含了惟一标识各州的数字, name 中的字符串表示州名, flag 中的字符串给出了州旗图形的文件名(如 "flag1.png"), capital 中的字符串表示州政府, flower 中的字符串表示州花, tree 中的字符串表示州树, bird 中的字符串则表示州鸟。
- a) **打开 states.jsp** 打开练习 30.12 中创建的 states.jsp 文件。
 - b) **从 java.sql 中导入供 JSP 使用的类** 将图 31.21 中第 4 行至第 5 行插入 states.jsp 中的第 4 行。
 - c) **添加 JSP 脚本** 在 states.jsp 源代码中 HTML select 元素的内部添加一段 JSP 脚本, 内含一个 try 语句块和一个捕获从 try 语句块中抛出 SQLException 异常的 catch 语句块。
 - d) **添加一个针对数据库的 Connection** 在上一步中定义的 try 语句块的内部, 利用值 "C:\Examples\Tutorial31\Exercises\Databases" 指定数据库的位置, 载入数据库驱动程序类并连接 statefacts 数据库(jdbc:db2j:statefacts)(注意: 如果教程 31 的位置并非 C:\Examples, 那么请将 C:\Examples 目录替换为教程 31 所在的目录)。
 - e) **创建 Statement 执行从数据库中检索州名的查询** 在 try 语句块的内部位于数据库连接成功代码的后面, 创建一个执行 SQL 查询的 Statement 对象。执行从数据库中获取所有州名的查询。
 - f) **关闭数据库连接** 执行完查询以后, 调用 Connection 对象的 close 方法断开同数据库的连接。

- g) 显示 SQLException 错误消息 在 catch 语句块的内部, 使用 JSP 隐含对象 out 的 println 方法显示 SQLException 异常的错误消息。
 - h) 保存文件 保存修改后的 states.jsp 文件。
 - i) 打开 stateFacts.jsp 打开习题 30.11 中创建的 stateFacts.jsp 文件。
 - j) 从 java.sql 中导入要使用的类 将图 31.21 中第 4 行至第 5 行插入 stateFacts.jsp 中的第 4 行。
 - k) 起始 JSP 脚本 在 stateFacts.jsp 源代码的内部, 位于 HTML 元素 h1 的后面, 起始一段 JSP 脚本。在该脚本的内部, 定义一个 try 语句块。
 - l) 添加一个针对数据库的 Connection 在 try 语句块的内部, 利用值 "C:\Examples\Tutorial31\Exercises\Databases" 指定数据库的位置, 以载入数据库驱动程序类并连接 statefacts 数据库 (注意: 如果教程 31 的位置并非 C:\Examples, 那么请使用教程 31 所在的目录替换 C:\Examples)。
 - m) 创建 Statement 执行从数据库中检索州信息的查询 在数据库连接成功代码的后面, 创建一个执行 SQL 查询的 Statement 对象。执行查询, 获取与所选州名相对应的州信息。
 - n) 结束脚本 在 HTML p 元素之前, 结束起始于步骤(k)的脚本, 这样, 便可放置 HTML 标记并实现对客户端的响应。
 - o) 添加另一个 JSP 脚本 在 stateFacts.jsp 源代码的内部, 位于 HTML p 元素中显示州鸟代码的后面, 再添加一段脚本, 实现关闭数据库连接、结束 try 语句块以及当 SQLException 异常发生时显示一条错误消息。
 - p) 保存文件 保存修改后的 stateFacts.jsp 文件。
 - q) 将 states.jsp 和 stateFacts.jsp 文件复制到 Tomcat webapps 目录下 将更新后的 states.jsp 和 stateFacts.jsp 文件复制到 C:\Program Files\ApacheGroup\Tomcat4.1\webapps\StateFacts 目录下。
 - r) 将数据库 JAR 文件复制到电话号码簿 Web 应用程序中 美国各州知识 Web 应用程序的信息层使用的是 Cloudscape 数据库。需要将 db2j.jar 和 license.jar 文件复制到 C:\Program Files\Apache Group\Tomcat 4.1\webapps\StateFacts\WEB-INF\lib 目录中。db2j.jar 和 license.jar 文件位于 C:\Cloudscape_5.1\lib 目录下。在教程 26 中, 如果 Cloudscape 的安装目录不是 C:\Cloudscape_5.1, 需将 C:\Cloudscape_5.1 目录替换为 Cloudscape 的安装目录。
 - s) 启动 Tomcat 选取 Start → Programs → Apache Tomcat 4.1 → Start Tomcat 启动 Tomcat 服务器。
 - t) 测试应用程序 打开一个 Web 浏览器, 分别输入 URL 地址 http://localhost:8080/StateFacts/states.jsp 和 http://localhost:8080/StateFacts/stateFacts.jsp 完成对应用程序的测试。
 - u) 停止 Tomcat 选取 Start → Programs → Apache Tomcat 4.1 → Stop Tomcat 停止 Tomcat 服务器。
- 31.13 (道路标志预览应用程序: 数据库) 利用 JSP 脚本创建道路标志预览应用程序数据库中的 Connection 对象和 Statement 对象。数据库 roadsigs 中有一张 signs 表, 内含 3 列: id, name 和 sign。其中, id 号中包含了惟一标路标的数字, name 中的字符串表示路标名称, sign 中的字符串给出了路标图形的文件名 (如 "sign01.png")。
- a) 打开 roadSigs.jsp 打开习题 30.13 中创建的 roadSigs.jsp 文件。
 - b) 从 java.sql 中导入供 JSP 使用的类 将图 31.21 中第 4 行至第 5 行插入 roadSigs.jsp 中的第 4 行。
 - c) 起始 JSP 脚本 在 roadSigs.jsp 源代码中位于 HTML h1 元素的后面, 添加一段 JSP 脚本。在该脚本的内部, 定义一个 try 语句块。
 - d) 添加一个针对数据库的 Connection 在 try 语句块的内部, 利用值 "C:\Examples\Tutorial31\Exercises\Databases" 指定数据库的位置, 载入数据库驱动程序类并连接 roadsigs 数据库 (jdbc:db2j:roadsigs) (注意: 如果教程 31 的位置并非 C:\Examples, 那么请将 C:\Examples 目录替换为教程 31 所在的目录)。
 - e) 创建 Statement 执行从数据库中检索路标信息的查询 在数据库连接成功代码的后面, 创建一个执行 SQL 查询的 Statement 对象。执行一条查询, 从数据库中获取所有路标信息。
 - f) 结束脚本 在 roadSigs.jsp 源代码的内部, 结束起始于步骤(c)的脚本, 这样, 便可放置 HTML 标记并实现对客户端的响应。

- g) **添加另一个 JSP 脚本** 在 roadSigns.jsp 源代码的内部, 位于 HTML img 元素中显示最后一幅路标图像代码的后面, 再添加一段脚本, 实现关闭数据库连接、结束 try 语句块以及当 SQLException 异常发生时显示一条的错误消息。
- h) **保存文件** 保存修改后的 roadSigns.jsp 文件。
- i) **将 roadSigns.jsp 文件复制到 Tomcat webapps 目录下** 将更新后的 roadSigns.jsp 文件复制到 C:\Program Files\Apache Group\Tomcat 4.1\webapps\RoadSign 目录下。
- j) **将数据库 JAR 文件复制到道路标志预览 Web 应用程序中** 道路标志预览 Web 应用程序的信息层使用的是 Cloudscape 数据库。需要将 db2j.jar 和 license.jar 文件复制到 C:\Program Files\Apache Group\Tomcat 4.1\webapps\RoadSign\WEB-INF\lib 目录中。db2j.jar 和 license.jar 文件位于 C:\Cloudscape_5.1\lib 目录下。在教程 26 中, 如果 Cloudscape 的安装目录不是 C:\Cloudscape_5.1, 需将 C:\Cloudscape_5.1 目录替换为 Cloudscape 的安装目录。
- k) **启动 Tomcat** 选取 Start → Programs → Apache Tomcat 4.1 → Start Tomcat 启动 Tomcat 服务器。
- l) **测试应用程序** 打开一个 Web 浏览器, 输入 URL 地址 <http://localhost:8080/RoadSign/roadSigns.jsp> 完成对应用程序的测试。
- m) **停止 Tomcat** 选取 Start → Programs → Apache Tomcat 4.1 → Stop Tomcat 停止 Tomcat 服务器。

教程 32 Web 书店应用程序：中间层

介绍 form 属性：method/action 和

在 JSP 中插入查询结果集



教学目标

在本教程中，读者将学到以下内容：

- 使用 JSP 实现中间层功能
- 处理 JSP 脚本中的 ResultSet
- 在 JSP 页面中插入 JSP 表达式

通过教程 30 和教程 31 的学习，我们构建了 Web 书店应用程序的客户层，并实现了同信息层的连接。利用 HTML 元素，设计了这个基于 Web 的应用程序的用户界面。在本教程中，将学习有关中间层的概念，之后，完成整个 Web 书店应用程序中间层功能的编写。前面曾经讲过，中间层负责客户层和信息层之间的交互工作，它接收来自客户层的用户请求，从信息层（即数据库）中检索数据并最终以 Web 形式响应用户。

32.1 回顾 Web 书店应用程序

我们曾经对三层 Web 书店应用程序进行过探讨（参见教程 29），并创建了该应用程序的 HTML 元素（参见教程 30）和数据库对象（参见教程 31）。现在，将编写实现 Web 书店应用程序功能的代码，完成客户层与数据库层之间的交互。这意味着我们编写的代码能够决定页面上将显示哪一本书，以及哪些信息需要从数据库中检索并显示出来。与此同时，还将编写当 View Information 按钮被点击时，能够把客户浏览器重定向到另外一个页面上。在开始这些任务以前，首先回顾一下教程 30 中的 ACE 表及伪代码，其中列出了完成该应用程序所需的操作、组件及事件。

32.2 在 books.jsp 页面中添加功能

尽管我们已设计了 Web 书店应用程序的 GUI 并建立了与数据库之间的连接，然而该书店应用程序目前仍未实现其相关功能，如列出所有图书的书名、点击 View Information 按钮时客户端浏览器的重定向，或者显示用户所指定图书相关信息等内容。下面，将开始编写实现这些功能的代码。

在 books.jsp 页面中显示书名

1. 将模板复制到工作目录中 将 C:\Examples\Tutorial32\TemplateApplication\bookstore_middleTier 目录复制到 C:\SimplyJava 目录中。
2. 打开 books.jsp 模板文件 在文本编辑器中打开 books.jsp 模板文件。
3. 获取书名 将图 32.1 中第 56 行至第 62 行添加到 JSP 页面中。完成下一步之后，此 while 语句显示数据库中每本书的书名。第 59 行至第 60 行把从 results（ResultSet 类型）中获取的书名赋值给 String 型变量 currentTitle。

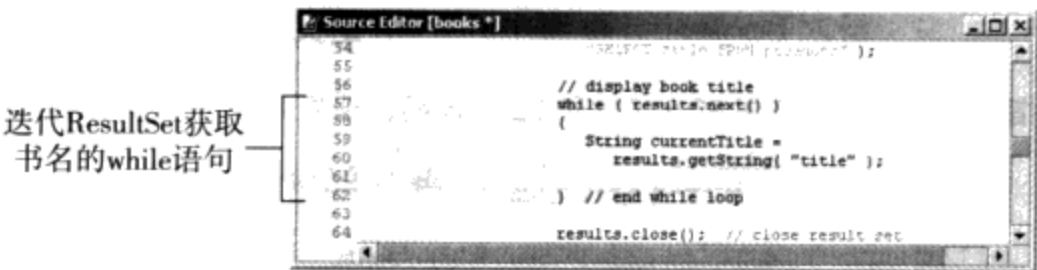


图 32.1 从 ResultSet 中获取书名的 while 语句

4. 显示书名 将图 32.2 中第 62 行至第 67 行添加到开始于第 56 行的 while 循环中。为了能将插入的 HTML 标记作为 while 循环的一部分，需在第 62 行结束此段脚本。第 65 行所使用的 HTML option 元素应位于 HTML select 元素的内部，目的是将书名添加到 HTML 菜单控件中。图 32.4 显示了当书名添加至菜单控件以后的运行结果。option 元素用于将选项添加到由 select 元素定义的菜单控件中。这里使用的 JSP 表达式可实现在 option 元素中插入书名。JSP 表达式位于<%=和%>之间，用来在 Web 页面中添加动态内容（如取自数据库中的信息）。在第 65 行，通过表达式<%=currentTitle%>插入 currentTitle 的值以响应客户请求。第 67 行又将恢复脚本，使之可以继续 JSP 页面中插入 Java 代码。

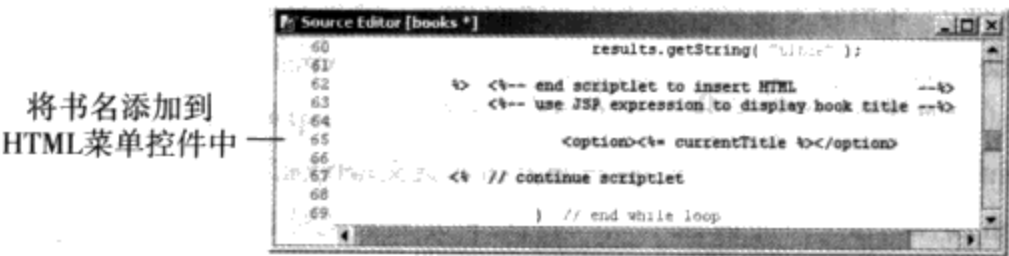


图 32.2 显示当前书名

5. 关闭 ResultSet 将图 32.3 中第 71 行添加到 JSP 页面中。第 71 行调用 ResultSet 的 close 方法释放由 results 所引用的资源，如 ResultSet 中的数据行资源。

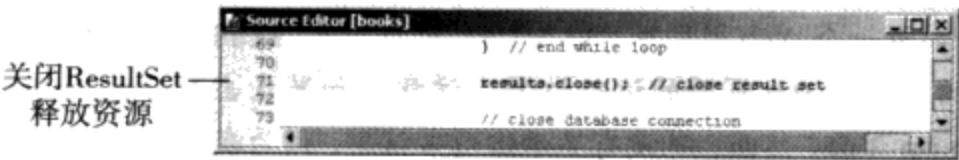


图 32.3 关闭 ResultSet

- 6. 保存应用程序 保存修改后的源代码文件。
- 7. 将 books.jsp 复制到 bookstore 目录中 将更新后的 books.jsp 文件复制到 C:\Program Files\Apache Group\Tomcat 4.1\webapps\bookstore 目录下。
- 8. 启动 Tomcat 选取 Start → Programs → Apache Tomcat 4.1 → Start Tomcat 启动 Tomcat 服务器。
- 9. 测试应用程序 打开一个 Web 浏览器，输入 URL 地址 http://localhost:8080/bookstore/books.jsp。图 32.4 显示了更新后的 books.jsp 页面的运行结果。注意，此时的 HTML 菜单控件中已装有书名。如果希望看到由 option 元素生成的 HTML，点击浏览器窗口中的查看菜单并选择源文件。
- 10. 关闭浏览器 点击浏览器窗口的关闭按钮关闭浏览器。
- 11. 停止运行 Tomcat 选取 Start → Programs → Apache Tomcat 4.1 → Stop Tomcat 停止 Tomcat 服务器。

下面，将定义 books.jsp 页面中的 HTML form（表单）操作。当点击 View Information 按钮时（如图 32.4 所示），HTML form 会将用户浏览器重定位至 bookInformation.jsp 页面中。

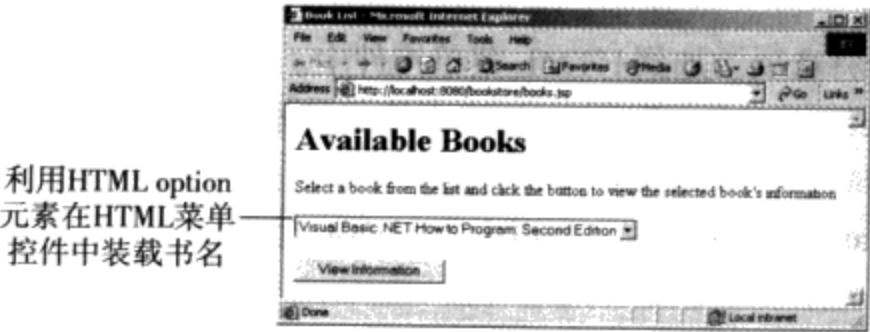


图 32.4 运行更新后的 books.jsp

针对 books.jsp 页面中的 form 定义 action

1. 为 form 元素指定 action 参照图 32.5 修改第 22 行代码, 设置 form 元素的 method 属性和 action 属性。

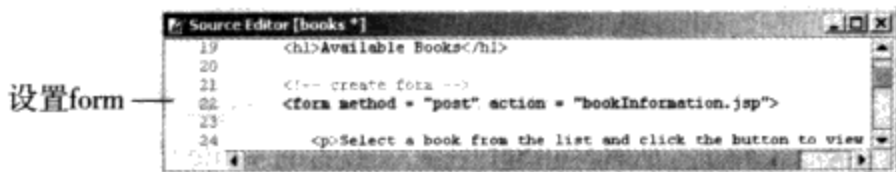


图 32.5 为 form 元素添加 action

form 元素的 method 属性指定表单内数据如何发送到 Web 服务器中。method = "post" 表示将用户的输入 (如用户选择的书名) 追加到客户端浏览器请求中。一旦收到客户端浏览器的请求, 位于 Web 服务器上的 JSP 便可访问作为请求部分的表单数据。例如, 当用户选择某一本书时, 参数 bookTitle (标识 select 元素的变量) 的值便被设置为所选书的书名, 这一信息同时会被添加到 request 对象中, 因而可在 bookInformation.jsp 页面中使用它们。当 bookInformation.jsp 收到客户端浏览器的请求时, 使用 request 对象的 getParameter 方法 (如图 32.7 中第 20 行), 访问作为请求部分而被发送的 bookTitle 参数的值。

form 元素的 action 属性指定当用户提交表单时 (点击 View Information 按钮), 表单数据 (用户输入) 的发往目的地。这里 (参见第 22 行), 指定一个名为 bookInformation.jsp 的 JSP 页面显示所选图书的信息。当用户点击 View Information 按钮时, HTML form 会将用户输入提交给 bookInformation.jsp 页面, 而此时 bookInformation.jsp 应作为 action 属性的值 (参见第 22 行), 利用这一方式实现对 bookInformation.jsp 页面的请求。

2. 保存应用程序 保存修改后的源代码文件。
3. 将 books.jsp 复制到 bookstore 目录中 将更新后的 books.jsp 文件复制到 C:\Program Files\Apache Group\Tomcat 4.1\webapps\bookstore 目录中。
4. 启动 Tomcat 选取 Start → Programs → Apache Tomcat 4.1 → Start Tomcat 启动 Tomcat 服务器。
5. 测试应用程序 打开一个 Web 浏览器, 输入 URL 地址 http://localhost:8080/bookstore/books.jsp。图 32.6 显示了更新后的 books.jsp 页面的运行结果。注意, 如果点击 View Information 按钮, 会载入 bookInformation.jsp 页面。然而, 并没有显示出图书信息, 因为我们尚未在 bookInformation.jsp 中添加此项功能。

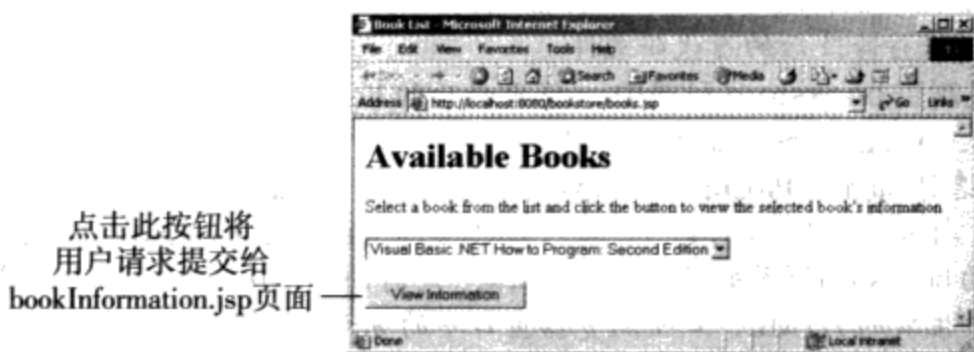


图 32.6 运行更新后的 books.jsp 页面

6. 关闭浏览器 点击浏览器窗口的关闭按钮关闭浏览器。
7. 停止 Tomcat 选取 Start → Programs → Apache Tomcat 4.1 → Stop Tomcat 停止 Tomcat 服务器。

自测题

1. _____ 元素用于将选项添加到 HTML 菜单控件中。
a) select b) item c) option d) 以上答案都不对
2. 当用户提交表单时, form 元素的 _____ 属性可指出所执行的任务。
a) action b) data c) url d) 以上答案都不对

答案: 1) c 2) a

32.3 在 bookInformation.jsp 页面中添加功能

下面，将在 bookInformation.jsp 页面中添加相应的代码，实现从数据库中检索用户请求的图书信息。

在 bookInformation.jsp 页面中显示图书的信息

- 1. 打开 bookInformation.jsp 模板文件 在文本编译器中打开 bookInformation.jsp 模板文件。
- 2. 在 h1 元素中显示书名 参照图 32.7 修改第 21 行。第 21 行利用一个 JSP 表达式从 JSP 隐含对象 request 中获取 bookTitle 参数的值，并将该值置于 h1 元素中。注意，bookTitle 参数对应菜单控件名（如图 30.6 中的第 25 行所示）。

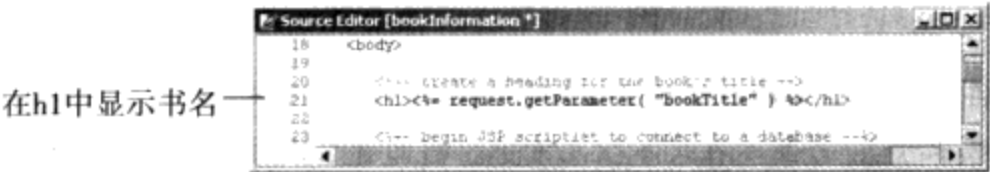


图 32.7 显示书名

- 3. 访问 results 将图 32.8 中第 52 行添加到 JSP 页面中。第 52 行将光标移动至 ResultSet 对象（results）的第一行上。

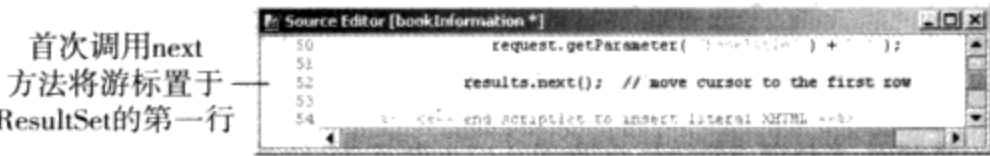


图 32.8 访问 results

- 4. 显示封面图片 将模板文件中第 57 行替换为图 32.9 中第 57 行至第 59 行的代码。第 57 行至第 59 行使用两个 JSP 表达式<%= results.getString("cover") %>和<%= results.getString("title") %>分别检索数据库中封面图片的文件名和书名。图片文件名会被追加到 "images/" 中，然后赋值给 img 元素的 src 属性。所有的图片文件均位于 C:\Program Files\Apache Group\Tomcat 4.1\webapps\bookstore\images 目录下。书名被追加到 "Book cover for" 中，然后赋值给 img 元素的 alt 属性。

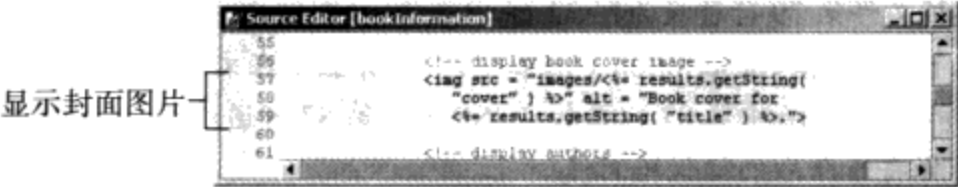


图 32.9 显示封面图片

- 5. 显示作者信息 将模板文件第 62 行替换为图 32.10 中第 62 行至第 63 行的代码。第 62 行至第 63 行使用一个 JSP 表达式从 results 获取作者（authors）信息，并显示在文本段元素中。

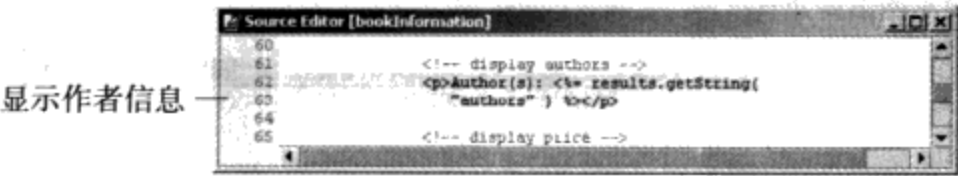


图 32.10 显示作者信息

- 6. 显示图书价格 将模板文件第 66 行替换为图 32.11 中第 66 行的代码。第 66 行使用一个 JSP 表达式从 results 获取图书的价格信息，并显示在文本段元素中。

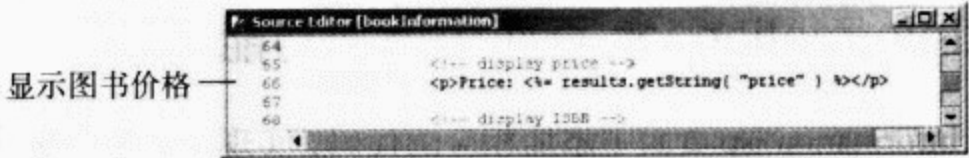


图 32.11 显示图书价格

7. 显示 ISBN 编码 将模板文件第 69 行替换为图 32.12 中第 69 行的代码。第 69 行使用一个 JSP 表达式从 results 获取 ISBN 编码，并显示在文本段元素中。

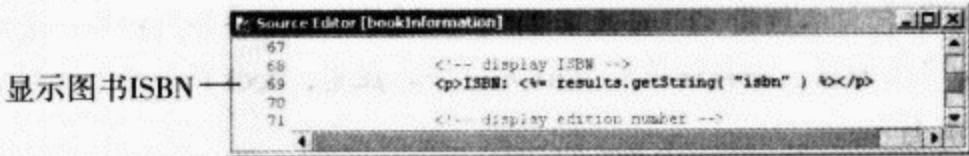


图 32.12 显示 ISBN

8. 显示版本号 将模板文件第 72 行替换为图 32.13 中第 72 行的代码。第 72 行使用一个 JSP 表达式从 results 获取 edition（版本号），并显示在文本段元素中。前面曾经讲过，使用 getInt 方法能够返回 edition 列的 int 值。

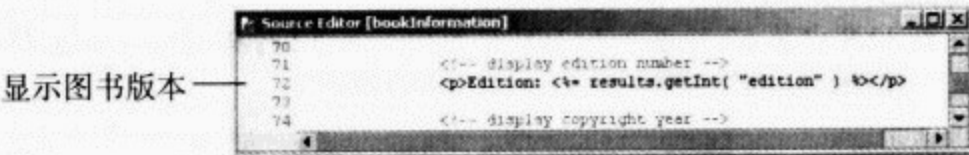


图 32.13 显示版本

9. 显示版权年限 将模板文件第 75 行替换为图 32.14 中第 75 行至第 76 行的代码。第 75 行至第 76 行使用一个 JSP 表达式 results 获取 copyrightYear（版权年限），并显示在文本段元素中。

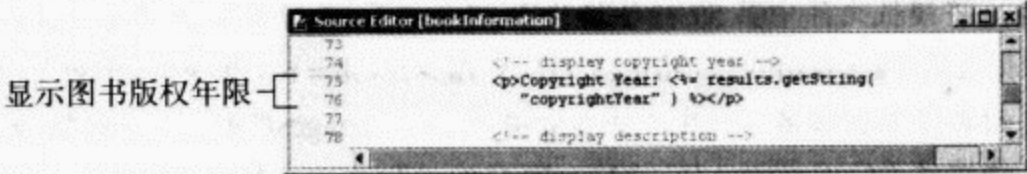


图 32.14 显示版权年限

10. 显示简介 将模板文件第 79 行替换为图 32.15 中第 79 行至第 80 行的代码。第 79 行至第 80 行使用一个 JSP 表达式从 results 获取 description（图书简介），并显示在文本段元素中。
11. 关闭 ResultSet 将图 32.16 中第 87 行添加到 JSP 中。第 87 行通过 close 方法关闭 results。
12. 保存应用程序 保存修改后的源代码文件。
13. 将 bookInformation.jsp 复制到 bookstore 目录中 将更新后的 bookInformation.jsp 文件复制到 C:\ProgramFiles\Apache Group\Tomcat 4.1\webapps\bookstore 目录下。

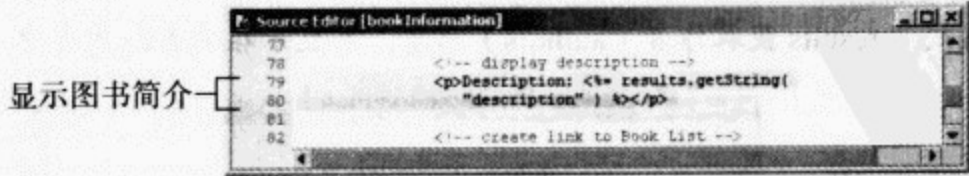


图 32.15 显示简介

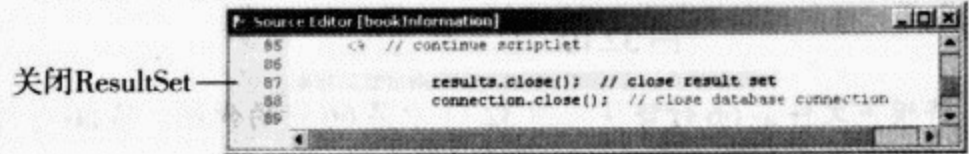


图 32.16 关闭 ResultSet

14. 启动 Tomcat 选取 Start → Programs → Apache Tomcat 4.1 → Start Tomcat 启动 Tomcat 服务器。
15. 测试应用程序 打开一个Web浏览器,输入URL地址http://localhost:8080/bookstore/books.jsp。从HTML菜单控件中选择"Visual Basic.NET How to Program:Second Edition"。图 32.17 显示了完成后的bookInformation.jsp页面的运行结果。可以看到,此时bookInformation.jsp页面中显示了所选图书的相关信息。

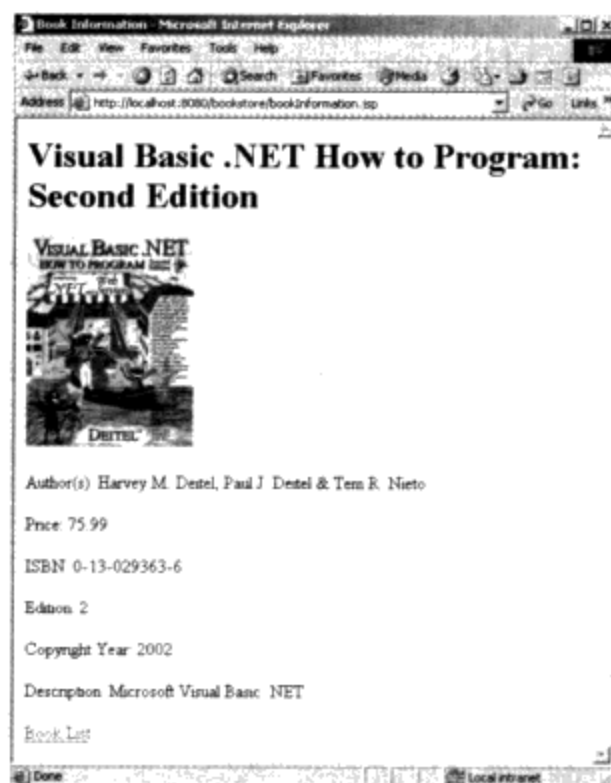


图 32.17 显示被选图书信息的 bookInformation.jsp 页面 (该书封面由 Prentice Hall 出版社提供)

16. 关闭浏览器 点击浏览器窗口的关闭按钮关闭浏览器。
17. 停止 Tomcat 选取 Start → Programs → Apache Tomcat 4.1 → Stop Tomcat 停止 Tomcat 服务器。

图32.18和图32.19中分别显示了完成后的Web书店应用程序中的books.jsp页面和bookInformation.jsp页面的源代码。随着JSP页面的进一步实现,这个三层Web书店应用程序便完成了。

```

1 <!-- Tutorial 32: books.jsp -->
2 <!-- Displays a form. -->
3
4 <%-- import java.sql.* for database classes --%>
5 <%@ page import = "java.sql.*" %>
6
7 <!-- begin HTML document -->
8 <html>
9
10 <!-- specify HTML head element -->
11 <head>
12
13 <!-- specify page title -->
14 <title>Book List</title>
15 </head>
16
17 <!-- begin body of document -->
18 <body>
19 <h1>Available Books </h1>
20
21 <!-- create form -->
22 <form method = "post" action = "bookInformation.jsp"> 设置 form 中的 action
23

```

```

24      <p>Select a book from the list and click the button to view
25      the selected book's information</p>
26
27      <!-- create list that contains book titles -->
28      <select name = "bookTitle">
29
30          <%-- begin JSP scriptlet to connect to database --%>
31          <%
32              // setup database connection
33              try
34              {
35                  // specify database location
36                  System.setProperty( "db2j.system.home",
37                      "C:\\Examples\\Tutorial29\\Databases" );
38
39                  // load Cloudscape driver
40                  Class.forName( "com.ibm.db2j.jdbc.DB2jDriver" );
41
42                  // connect to database
43                  Connection connection =
44                      DriverManager.getConnection(
45                          "jdbc:db2j:bookstore" );
46
47                  // obtain list of titles
48                  if ( connection != null )
49                  {
50                      Statement statement =
51                          connection.createStatement();
52
53                      ResultSet results = statement.executeQuery(
54                          "SELECT title FROM products" );
55
56                      // display book title
57                      while ( results.next() )
58                      {
59                          String currentTitle = results.getString( "title" );
60                          // 从ResultSet 中获取书名
61
62                      } <%-- end scriptlet to insert HTML --%>
63                      <%-- JSP expressions output from this loop --%>
64
65                      <option><%= currentTitle %></option>
66                      // 将书名添加到HTML菜单控件中
67                  <% // continue scriptlet
68
69                      } // end while loop
70
71                      results.close(); // close result set
72                      // 关闭 ResultSet
73
74                      // close database connection
75                      connection.close();
76
77                  } // end if
78
79              } // end try
80
81              // catch SQLException
82              catch ( SQLException exception )
83              {
84                  out.println(

```

```

84             "Exception: " + exception + " occurred." );
85         }
86
87         %> <!-- end scriptlet --%>
88
89     </select>
90
91     <!-- create View Information button -->
92     <p><input type = "submit" value = "View Information"></p>
93 </form>
94
95 </body>
96 </html>

```

图 32.18 books.jsp 源代码

```

1 <!-- Tutorial 32: bookInformation.jsp -->
2 <!-- Displays book information. -->
3
4 <!-- import java.sql.* for database classes --%>
5 <%@ page import = "java.sql.*" %>
6
7 <!-- begin HTML document -->
8 <html>
9
10 <!-- specify head element -->
11 <head>
12
13 <!-- specify page title -->
14 <title>Book Information</title>
15 </head>
16
17 <!-- begin body of document -->
18 <body>
19
20 <!-- create a heading for the book's title -->
21 <h1><%= request.getParameter( "bookTitle" ) %></h1>
22                                     将books.jsp中选定的书名显示在h1中
23 <!-- begin JSP scriptlet to connect to a database --%>
24 <%
25     // setup database connection
26     try
27     {
28         // specify database location
29         System.setProperty( "db2j.system.home",
30             "C:\\\\Examples\\\\Tutorial29\\\\Databases" );
31
32         // load Cloudscape driver
33         Class.forName( "com.ibm.db2j.jdbc.DB2jDriver" );
34
35         // obtain connection to database
36         Connection connection = DriverManager.getConnection(
37             "jdbc:db2j:bookstore" );
38
39         // obtain list of titles from database
40         if ( connection != null )
41         {
42             // create statement
43             Statement statement = connection.createStatement();
44
45             // execute query to get book information

```



```

46         ResultSet results = statement.executeQuery(
47             "SELECT cover, title, authors, price, isbn, " +
48             "edition, copyrightYear, description " +
49             "FROM products WHERE title = '" +
50             request.getParameter( "bookTitle" ) + "' );
51
52         results.next(); // move cursor to the first row
53                                     移动至 ResultSet 的第一行
54     %> <%-- end scriptlet to insert literal HTML --%>
55
56         <!-- display book cover image -->
57         <img src = "images/<%= results.getString(
58             "cover" ) %>" alt = "Book cover for
59             <%= results.getString( "title" ) %>.">
60                                     显示封面图片
61
62         <!-- display authors -->
63         <p>Author(s): <%= results.getString(
64             "authors" ) %></p>
65                                     显示作者
66
67         <!-- display price -->
68         <p>Price: <%= results.getDouble( "price" ) %></p>
69                                     显示价格
70
71         <!-- display ISBN -->
72         <p>ISBN: <%= results.getString( "isbn" ) %></p>
73                                     显示 ISBN
74
75         <!-- display edition number -->
76         <p>Edition: <%= results.getInt( "edition" ) %></p>
77                                     显示版本号
78
79         <!-- display copyright year -->
80         <p>Copyright Year: <%= results.getString(
81             "copyrightYear" ) %></p>
82                                     显示版权年限
83
84         <!-- display description -->
85         <p>Description: <%= results.getString(
86             "description" ) %></p>
87                                     显示图书简介
88
89         <!-- create link to Book List -->
90         <p><a href = "books.jsp"> Book List</a></p>
91
92     <% // continue scriptlet
93
94         results.close(); // close result set
95                                     关闭 ResultSet
96         connection.close(); // close database connection
97
98     } // end if
99
100 } // end try
101
102 // catch SQLException
103 catch ( SQLException exception )
104 {
105     out.println( "Exception: " + exception + " occurred." );
106 }
107
108 %> <%-- end scriptlet --%>
109
110 </body>
111 </html>

```

图 32.19 bookInformation.jsp 源代码

自测题

1. 针对返回的 _____ 执行 SQL 查询可以检索数据库中的信息。
a) ResultSet b) Connection
c) Statement d) 以上答案都不对
2. 为了从 ResultSet 对象 results 中获取 "title" 列的 String 型表示, 需使用 _____ 语句。
a) ResultSet.getString("title"); b) results.getString("title");
c) ResultSet.getValue("title"); d) results.getValue("title");

答案: 1) a 2) b

32.4 Internet 与 Web 资源

让我们停留片刻, 访问一些网站。为了节省时间, 可以直接通过随书附带的 CD-ROM 或 www.deitel.com 链接这些热门站点。

java.sun.com/products/jsp/

该网站全面介绍 JSP 并提供了与 JSP 相关的教程, 同时还包含了与 JSP 相关的新闻和要点的链接。

java.sun.com/products/jsp/docs.html

该网站提供了许多教程、文章、代码实例以及与 JSP 有关的链接资源。

www.jspin.com/home/tutorials

该网站为用户提供了许多 JSP 教程及代码实例。

www.jsp-servlet.net/tomcat_examples.html

该网站提供了许多免费的 JSP 案例及源代码。

32.5 小结

本教程中, 我们编写了三层 Web 书店应用程序的中间层。通过添加 HTML form 元素的 method 属性和 action 属性及 JSP 脚本与表达式, 可以指定用户同 JSP 页面交互时所执行的操作。同时还学习了 JSP 表达式, 以及如何在 JSP 页面内使用它们显示相关内容。form 元素的 method 属性, 表示表单中的数据如何发送到 Web 服务器中。利用 form 元素的 action 属性, 可以将客户端浏览器重定向到其他 JSP 页面上。

在学完 JSP 表达式、HTML form 元素的 method 和 action 属性以后, 将它们应用到了 Web 书店应用程序中。本教程从应用程序的第一个 JSP 页面 books.jsp 开始, 当页面被加载时, 会从数据库中获取书名并把它们显示在 HTML 菜单控件中。这是通过在 HTML option 元素中添加 JSP 表达式来完成的。之后, 还定义了点击 View Information 按钮时需完成的一些操作。在 form 元素中, 将 method 属性设置为 post 能够把用户输入追加到客户端浏览器的请求中, 而设置 action 属性则能够把用户浏览器中的 books.jsp 页面重定向到 bookInformation.jsp 页面中。

随后, 定义了 bookInformation.jsp 页面, 利用所添加的 JSP 表达式显示被选图书的相关信息。使用隐含对象 request 的值可以确定所选图书的书名, 经检索的图书将最终被显示在 HTML p(文本段) 中。这样, 利用编程便可以控制信息层到客户层的数据流并最终完成整个三层 Web 书店应用程序的开发过程。

技术小结

在 JSP 中添加 JSP 表达式

- 将表达式放置于 <%= 和 %> 之间。

获取客户请求

- 使用隐含对象 request 的 getParameter 方法从客户请求中获取数据。

将客户端浏览器重定向至其他 Web 页面

- 当用户提交表单时，利用 form 元素的 action 属性将客户请求重定向到另外一个 Web 页面上。

使用 JSP 和 JDBC 创建一个三层 Web 应用程序

- 在 JSP 中放置 HTML 元素和控件创建 Web 应用程序的客户层。
- 使用数据库（如 Cloudscape）实现 Web 应用程序的信息层。
- 使用 JSP 技术（如脚本及表达式）为 Web 应用程序创建商业逻辑层功能。

关键术语

HTML form 元素的 action 属性 当点击 "submit" 按钮提交表单时，此属性指出了表单数据发送的目的地。

JSP 表达式 位于 <%= 和 %> 之间的表达式。JSP 表达式用于插入能够响应客户的动态内容（如检索数据库中的信息）。

HTML form 元素的 method 属性 该属性描述了 HTML 表单中的数据是如何发送到 Web 服务器中的。

HTML option 元素 此元素能够将选项添加到 HTML 菜单控件中（由 HTML select 元素定义）。

HTML 参考索引

option 此 HTML 元素位于 HTML select 元素的内部，用于将选项添加到菜单控件中。

form 此 HTML 元素用于在 HTML 文档中插入一个收集用户输入的表单。

● 属性

action 当点击 "submit" 按钮提交表单时，指明表单数据发送的目的地。

method 描述 HTML 中的表单数据将如何发送到 Web 服务器中。

JSP 参考索引

JSP 表达式 该表达式通过插入动态内容（如检索数据库中的信息）完成对客户端请求的响应。JSP 表达式位于 <%= 和 %> 之间。

习题

选择题

32.1 一个 JSP 表达式可以 _____。

- a) 将客户端浏览器重定向到其他不同的 Web 页面上
- b) 定义按钮点击时的功能
- c) 在 Web 网页中添加动态内容
- d) 定义选择 Web 控件时的功能

32.2 HTML form 元素的 action 属性 _____。

- a) 刷新当前的 Web 网页
- b) 响应用户的输入
- c) 指定用户提交表单时将要执行的任务
- d) 对 HTML 菜单控件的选择做出响应

32.3 JSP 隐含对象 _____ 用来在 Web 书店应用程序中检索客户请求的数据。

- a) out
- b) request
- c) clientRequest
- d) 以上答案都不对

32.4 ResultSet 对象用于 _____。

- a) 接收特定用户的数据 b) 从数据库中检索数据
 - c) 连接数据库 d) 以上答案都不对
- 32.5 JSP 表达式需放置在 _____ 之间。
- a) <%-- 和 --%> b) <% 和 %> c) <%= 和 %> d) <%= 和 =%>
- 32.6 form 元素的 method 属性 _____。
- a) 指明当用户点击 "submit" 按钮时将调用的方法
 - b) 描述表单数据将如何发送到 Web 服务器中
 - c) 将客户端浏览器重定向到其他 Web 页面上
 - d) 以上答案都不对
- 32.7 request.getParameter 方法需接收一个 _____ 作为参数。
- a) URL b) int c) boolean d) String
- 32.8 _____ 属性指出了 img 元素所显示图片的位置。
- a) src b) imageURL c) image d) display
- 32.9 _____ 元素用于将选项添加到 HTML 菜单控件中。
- a) item b) option c) value d) 以上答案都不对
- 32.10 JSP 表达式 _____ 能够把传递给 JSP 页面的 name 作为对客户端请求的一部分进行显示。
- a) <%= request.getName()%> b) <%= request.getPara("name")%>
 - c) <%= request.getParameter("name")%> d) 以上答案都不对

练习题

- 32.11 (电话号码簿应用程序：功能实现) 定义电话号码簿应用程序的中间层。
- a) 打开 phoneBook.jsp 打开经习题 31.11 中修改过的 phoneBook.jsp 文件。
 - b) 在 phoneBook.jsp 页面中定义 form 的 method 属性和 action 属性 在 phoneBook.jsp 源代码中，将 form 元素的 method 属性设定为 "post"，利用 action 属性将客户端请求提交给 phoneNumber.jsp 页面。
 - c) 利用姓名装载 HTML 菜单控件 在 phoneBook.jsp 源代码中，添加一段脚本，其内有一条 while 语句。此循环将把每个人的姓名(取自 ResultSet) 添加到 HTML 菜单控件中。该步骤需要 HTML 标记及一个 JSP 表达式。
 - d) 关闭 ResultSet 在结束 while 语句脚本的内部，调用 ResultSet 的 close 方法关闭该 ResultSet。
 - e) 保存文件 保存修改后的源代码文件。
 - f) 打开 phoneNumber.jsp 打开经习题 31.11 中修改后的 phoneNumber.jsp 文件。
 - g) 显示被选姓名 在 phoneNumber.jsp 源代码中，修改 h1 元素，将接收到的姓名作为对客户端的请求部分进行显示。
 - h) 显示电话号码 在 phoneNumber.jsp 源代码中，修改包含 "numbers" 的 HTML p 元素，显示与所选姓名相对应的电话号码。注意，为获取电话号码需要将游标移动至 ResultSet 的第一行。
 - i) 关闭 ResultSet 在步骤(h)添加代码的后面(位于脚本内)，调用 ResultSet 的 close 方法关闭该 ResultSet。
 - j) 保存文件 保存修改后的源代码文件。
 - k) 将 phoneBook.jsp 和 phoneNumber.jsp 复制到 PhoneBook 目录下 将更新后的 phoneBook.jsp 文件和 phoneNumber.jsp 文件复制到 C:\Program Files\Apache Group\Tomcat4.1\webapps\PhoneBook 目录下。
 - l) 启动 Tomcat 选取 Start → Programs → Apache Tomcat 4.1 → Start Tomcat 启动 Tomcat 服务器。
 - m) 测试应用程序 打开一个 Web 浏览器，输入 URL 地址 http://localhost:8080/PhoneBook/PhoneBook.jsp，从列表选择一个姓名并点击 Get Number 按钮完成对应用程序的测试。
 - n) 停止 Tomcat 选取 Start → Programs → Apache Tomcat 4.1 → Stop Tomcat 停止 Tomcat 服务器。

32.12 (美国各州知识应用程序: 功能实现) 定义美国各州知识应用程序的中间层。

- a) 打开 states.jsp 打开经习题 31.12 修改后的 states.jsp 文件。
- b) 在 states.jsp 页面中定义 form 的 method 属性和 action 属性 在 states.jsp 源代码中, 将 form 元素的 method 属性设定为 "post", 利用 action 属性将客户请求提交给 stateFacts.jsp 页面。
- c) 利用州名装载 HTML 菜单控件 在 states.jsp 源代码中, 添加一段脚本, 内含一条 while 语句。此循环将把每个州的名称 (取自 ResultSet) 添加到 HTML 菜单控件中。该步骤需要 HTML 标记及一个 JSP 表达式。
- d) 关闭 ResultSet 在脚本内部, 调用 ResultSet 的 close 方法关闭该 ResultSet。
- e) 保存文件 保存修改后的源代码文件。
- f) 打开 stateFacts.jsp 打开经习题 31.11 修改后的 stateFacts.jsp 文件。
- g) 显示所选州的州名 在 stateFacts.jsp 源代码中, 修改含 "State Name" 的 h1 元素, 显示所选州的州名。
- h) 移至 ResultSet 的第一行 在包含 ResultSet 的脚本内部, 调用 ResultSet 的 next 方法将光标移动至 ResultSet 的第一行上。
- i) 显示州旗 在 stateFacts.jsp 源代码中, 修改 img 元素显示所选州的州旗。
- j) 显示州政府 在 stateFacts.jsp 源代码中, 修改含 "Capital:" 的 p 元素, 显示所选州的州政府。
- k) 显示州花 在 stateFacts.jsp 源代码中, 修改含 "Flower:" 的 p 元素, 显示所选州的州花。
- l) 显示州树 在 stateFacts.jsp 源代码中, 修改含 "Tree:" 的 p 元素, 显示所选州的州树。
- m) 显示州鸟 在 stateFacts.jsp 源代码中, 修改含 "Bird:" 的 p 元素, 显示所选州的州鸟。
- n) 关闭 ResultSet 在步骤(i)至步骤(m)所添代码的后面 (位于脚本内), 调用 close 方法关闭 ResultSet。
- o) 保存文件 保存修改后的源代码文件。
- p) 将 states.jsp 和 stateFacts.jsp 复制到 stateFacts 目录下 将更新后的 states.jsp 文件和 stateFacts.jsp 文件复制到 C:\Program Files\Apache Group\Tomcat 4.1\webapps\StateFacts 目录下。
- q) 启动 Tomcat 选取 Start → Programs → Apache Tomcat 4.1 → Start Tomcat 启动 Tomcat 服务器。
- r) 测试应用程序 打开一个 Web 浏览器, 输入 URL 地址 <http://localhost:8080/StateFacts/states.jsp>, 从列表中选择某个州名并点击 Review Facts 按钮完成对应用程序的测试。
- s) 停止 Tomcat 选取 Start → Programs → Apache Tomcat 4.1 → Stop Tomcat 停止 Tomcat 服务器。

32.13 (道路标志预览应用程序: 功能实现) 定义道路标志预览应用程序的中间层。

- a) 打开 roadSigns.jsp 打开经习题 31.13 修改后的 roadSigns.jsp 文件。
- b) 显示路标图片 在 roadSigns.jsp 源代码中, 添加一条 while 语句。此循环语句将从 ResultSet 的每一行中获取图片文件名及图片名, 并把它们插入页面中的 img 元素内。注意, 该循环将取代习题 30.13 中 JSP 页面所添加的 img 元素列表。其余的 img 元素将作为 HTML 标记位于起始和结束 while 语句之间的两个脚本内。利用两个 JSP 表达式从 ResultSet 中获取图片文件名和图片名。
- c) 关闭 ResultSet 在 while 语句的后面 (位于第二段脚本内), 调用 ResultSet 的 close 方法关闭此 ResultSet。
- d) 在 roadSigns.jsp 页面中定义 form 的 method 属性和 action 属性 在 roadSigns.jsp 源代码中, 将 form 元素的 method 属性设定为 "post", 利用 action 属性将客户端请求提交给 roadTestRegistered.jsp 页面。
- e) 保存文件 保存修改后的源代码文件。
- f) 打开 roadTestRegistered.jsp 打开习题 31.11 中创建的 roadTestRegistered.jsp 文件。
- g) 显示客户名字 在 roadTestRegistered.jsp 源代码中, 修改含 "confirmation" 的 p 元素, 使用一个 JSP 表达式从 JSP 隐含对象 request 中获取客户姓名并将它显示在文本段中。

- h) **显示客户电话号码** 修改上一步骤中的文本段元素，使用一个 JSP 表达式从 JSP 隐含对象 request 中获取客户的电话号码并将它显示在文本段中。
- i) **保存文件** 保存修改后的源代码文件。
- j) **将 roadSigns.jsp 和 roadTestRegistered.jsp 复制到 roadSigns 目录下** 将更新后的 roadSigns.jsp 文件和 roadTestRegistered.jsp 文件复制到 C:\Program Files\Apache Group\Tomcat 4.1\webapps\RoadSign 目录下。
- k) **启动 Tomcat** 选取 Start → Programs → Apache Tomcat 4.1 → Start Tomcat 启动 Tomcat 服务器。
- l) **测试该应用程序** 打开一个 Web 浏览器，输入 URL 地址 <http://localhost:8080/RoadSign/roadSigns.jsp>，录入客户姓名和电话号码，然后点击 Register 按钮完成对应用程序的测试。
- m) **停止 Tomcat** 选取 Start → Programs → Apache Tomcat 4.1 → Stop Tomcat 停止运行 Tomcat 服务器。

附录 A 运算符优先级表



以下运算符将按优先级高低从上到下进行排列，不同级别上的优先级以水平线为分隔线。注意，Java 运算符是按照从左到右的方向来结合的。

运算符	类型
++	单目后增
--	单目后减
++	单目先增
--	单目先减
+	单目加
-	单目减
!	单目逻辑非
~	单目按位反
(type)	单目造型
*	乘法
/	除法
%	取余
+	加法或字符串连接运算符
-	减法
<<	左移
>>	带符号右移
>>>	不带符号右移
<	小于
<=	小于等于
>	大于
>=	大于等于
instanceof	类型比较
==	相等
!=	不等
&	按位与 布尔逻辑与
^	按位异或 布尔逻辑异或
	按位或 布尔逻辑或
&&	条件与
	条件或
?:	条件运算符
=	赋值
+=	加赋值
-=	减赋值
*=	乘赋值
/=	除赋值
%=	取余赋值
&=	按位与赋值
^=	按位异或赋值
=	按位或赋值
<<=	按位左移赋值
>>=	按位带符号右移赋值
>>>=	按位不带符号右移赋值

图 A.1 运算符列表（按优先级顺序）



附录 B ASCII 字符集

图 B.1 中左侧一列的数字，为十进制字符码值（0~127）中左边数字位上的数字，该图最上方一行的数字是字符码值中最右边一位上的数字。例如，"F" 的字符码值为 70，"&" 的字符码值为 38。

使用本书的大多数读者，感兴趣的只是 ASCII 字符集如何通过计算机表示英文字符。ASCII 字符集是 Java 中使用的 Unicode 字符集的一个子集，Unicode 字符集能够表示世界上大多数语言使用的字符。

	0	1	2	3	4	5	6	7	8	9
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht
1	lf	vt	ff	cr	so	si	dle	dc1	dc2	dc3
2	dc4	nak	syn	etb	can	em	sub	esc	fs	gs
3	rs	us	sp	!	"	#	\$	%	&	'
4	()	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[\]	^	_	`	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{		}	~	del		

图 B.1 ASCII 字符集

附录 C GUI 设计导航



本附录包含每一教程末尾所列全部 GUI 设计导航。以下内容将按照教程顺序进行编排，各教程内部以组件为单位进行排列。

教程 2: Welcome 应用程序（引入图形用户界面的程序设计）

总体设计

- 可以在应用程序中使用不同颜色，但不应超出分散用户注意力的程度。

JFrame

- 选择简短的具备描述性质的 JFrame 标题。
- JFrame 的标题应使用书名大写形式，即每个重要单词的首字母应为大写，且不应使用任何标点符号作为结束。

JLabel

- 使用 JLabel 显示不允许用户修改的文本。
- 保证所有的 JLabel 组件能够足以显示它们的文本。
- 利用带有图片的 JLabel 可以改善需要使用一些不允许用户改动图形的 GUI。
- 保证所有的 JLabel 组件能够足以显示它们的图片。

教程 3: 库存清单应用程序（介绍 JTextField 和 JButton 组件）

JButton

- 使用 text 属性标注 JButton。JButton 上的文本应使用书名大写形式，并且在尽量简短的同时还可向用户表达一定的含义。
- 多个 JButton 应从 JFrame 的顶部位置垂直向下排列，或者是在 JFrame 底部位置的某一行上水平进行排列。

JFrame

- 将应用程序的输出组件置于应用程序输入组件的下方和 / 或右侧。

JLabel

- 利用 JLabel 标识其他 GUI 组件。
- 起描述性作用的 JLabel 应使用语句大写形式并以冒号结束。
- 将每一个起描述作用的 JLabel 放置在所标识组件（如 JTextField）的上方或者是左侧。
- 如果 JLabel 是垂直放置的，应将这一组起描述性作用的 JLabel 的左边界对齐。
- 如果组件按照水平方式进行放置，则起描述性作用的 JLabel 应该与它所描述的组件拥有相同的高度。
- 一个起描述性作用的 JLabel 和它所描述的组件，如果它们是以水平方式进行放置，则它们的顶部应该对齐。

- 一个起描述性作用的 JLabel 和它所描述的组件，如果它们是按照垂直方式进行放置，则它们均应靠左对齐。

JTextField

- 利用可编辑的 JTextField 从键盘中输入数据。默认情况下 JTextField 是可编辑的。
- 每一个 JTextField 应该有一个指明其意图的起描述性作用的 JLabel。
- 如果可能的话，为所期望的输入提供充足宽度的 JTextField。否则，文本会在用户输入时发生滚动并只显示出其中的一部分。
- 对齐垂直排列的 JTextField 的左边界。
- 一般来说 JTextField 中的数字应该靠右对齐。
- 作为输出的 JTextField 应该与作为输入的 JTextField 有所区别。将输出 JTextField 的 editable 属性设置为 false，防止用户向其输入数据，这同时也会使 JTextField 以灰色的背景出现在用户界面中。

教程 7：牙科付款应用程序（介绍 JCheckBox，消息对话框和逻辑运算符）

JCheckBox

- JCheckBox 的文本应具备描述性质且尽可能简练。同时，JCheckBox 文本还应采用书名大写形式。
- 将一组 JCheckBox 水平对齐或者垂直对齐。

消息对话框

- 消息对话框中所显示的文本应具备描述性质且尽可能简练。

教程 8：购车还贷计算器应用程序（介绍 while 循环语句及 JTextArea 组件）

JTextArea

- 在 JTextArea 中使用表头能够在显示列表数据时改善应用程序的可读性。

教程 9：班级平均分应用程序（介绍 do...while 循环语句）

JButton

- 若 JButton 的功能对用户来说是不可用的，则禁用该 JButton。
- 若 JButton 的功能对用户来说是可用的，则启用该 JButton。

JTextArea

- 大多数 JTextArea 都应配有一个描述其性质的 JLabel，用于指明所要显示的输出信息。对这样的 JLabel 使用语句大写形式。

教程 10：利息计算器应用程序（介绍 for 循环语句）

JTextArea

- 若某个 JTextArea 中需要按照多行形式显示输出，可将这一 JTextArea 同 JScrollPane 联系起来，使用户通过滚动方式看到 JTextArea 中的整个输出行。

JSpinner

- 当想使用 JTextField 但又希望用户在一个指定范围内输入有效数据时，可考虑使用 JSpinner。JSpinner 通过拒绝错误类型或者是超出某个范围的值来防止无效输入。当无效数据被拒绝以后，JSpinner 中以前曾显示的值将得到恢复。

- 为 JSpinner 和 JPasswordField 使用相同的 GUI 设计原则。

教程 11: 门禁系统应用程序 (介绍 switch 多向选择语句, Date 及 DateFormat 类)

总体设计

- 如果所开发的 GUI 被用于模拟现实世界中的某个物体, 则相应的 GUI 设计就应模拟出该物体的物理外观。

JPasswordField

- 利用 JPasswordField 屏蔽密码或其他敏感信息。

教程 16: 国旗知识测评应用程序 (介绍一维数组及 JComboBox 组件)

JComboBox

- 每一个 JComboBox 组件都应具有一个描述其内容的 JLabel。
- 对拥有多个选项的 JComboBox 进行排序将有助于用户查找到所需要的选项。

教程 17: 成绩评定应用程序 (介绍二维数组及 JRadioButton 组件)

JRadioButton

- 当用户只能从一组选项中选择出惟一的一个选项时, 可以考虑使用 JRadioButton 组件。
- 总是将属于同一组的 JRadioButton 放置在一个独立的 ButtonGroup 中。
- 对同一组中的 JRadioButton 按照水平或垂直方式对齐。

教程 19: 货运中心应用程序 (介绍集合, ArrayList 及迭代器)

总体设计

- 可利用助记符“点击”组件。助记符为用户提供了一种较为便利的方式, 尤其适用于需要用户输入大量文本数据的应用程序。
- 通过把 JList 加入到 JScrollPane 中可以使用户利用滚动条查看到 JList 中的所有条目。

教程 22: 打字训练器应用程序 (介绍键盘事件及 JMenu 组件)

菜单

- 菜单选项中的文本应使用书名大写形式。
- 如果点击某个菜单选项时会打开一个对话框, 则在该菜单选项文本的后面添加一个省略号。
- 利用分隔条分组管理 JMenu 中的多个 JMenuItem。

JCheckBoxMenuItem

- JCheckBoxMenuItem 中的文本应具有描述性质且尽可能简短, 同时还应使用书名大写形式。
- 当用户需要从菜单中选择多个选项时, 可以考虑使用 JCheckBoxMenuItem。

JRadioButtonMenuItem

- 当用户需要从菜单中选择惟一的一个选项时, 可以考虑使用 JRadioButtonMenuItem。



附录 D Java 类库索引

本附录包含正文中出现的所有组件及所使用到类库。对于每一个组件或使用到的类，配有与其相关的事件、属性及方法的说明。

教程 1: Moving Shapes 应用程序 (介绍计算机, Internet 及 Java 程序设计基础)

无新内容。

教程 2: Welcome 应用程序 (引入图形用户界面的程序设计)

JFrame 该组件能够让 Java 应用程序出现在窗口中。应用程序中的其他所有组件均显示在此应用程序窗口中。

- 运行



- 方法

setBackground 设置 JFrame 内容面板 (或者是其他组件) 的背景颜色。例如, 如果 JFrame 的内容面板为 `contentPane`, 则语句 `contentPane.setBackground(Color.YELLOW)`; 将把内容面板的背景色设置为黄色。

setSize 设置 JFrame 的大小 (以像素为单位)。

setTitle 设置 JFrame 标题栏中的文本。

JLabel 该组件显示不允许用户修改的文本和图片。

- 运行

Welcome to Java Programming!



- 方法

setBounds 指定 JLabel 的大小和位置。

setFont 指定 JLabel 中所显示文本的字体名、字形和字号大小。

setHorizontalAlignment 决定 JLabel 中文本的对齐方式。

setIcon 指定 JLabel 中图片的文件名和路径。

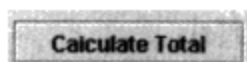
setSize 指定 JLabel 的高度和宽度 (以像素为单位)。

setText 指定 JLabel 中显示的文本。

教程 3: 库存清单应用程序 (介绍 JTextField 和 JButton 组件)

JButton 该组件可允许用户命令应用程序完成一项操作。

- 运行



● 方法

- setBounds 设置边界属性，利用其指明 JButton 的位置与大小。
- setText 设置 JButton 的文本属性。

JLabel 该组件用来显示不允许用户进行修改的文本。

● 运行

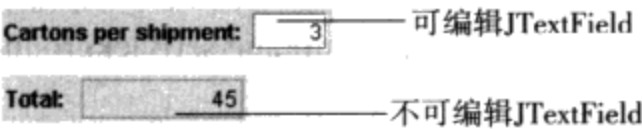


● 方法

- setBounds 设置边界属性，利用其指明 JLabel 的位置与大小。
- setFont 设置 JLabel 中所显示文本的字体名、字号及字形。例如，为设置 textJLabel 的字体使用语句 textJLabel.setFont(new Font(fontName,fontStyle,fontSize)); 其中属于 fontName 的一些实例包括 "SansSerif", "Serif" 和 "Monospaced"。fontStyle 可以是 Font.PLAIN, Font.BOLD, Font.ITALIC 或 Font.Bold + Font.ITALIC。fontSize 可以是任意正整数。
- setHorizontalAlignment 用于指定 JLabel 内文本的对齐方式(JLabel.LEFT, JLabel.CENTER, JLabel.RIGHT)。
- setIcon 为 JLabel 设置所要显示的图片。
- setLocation 设置 JLabel 的位置。
- setSize 设置 JLabel 的宽度和高度（以像素为单位）。
- setText 设置 JLabel 中显示的文本。

JTextField 该组件可以从键盘中获取输入或是将信息显示给用户。

● 运行



● 方法

- setBounds 设置边界属性，利用其指明 JTextField 的位置与大小。
- setEditable 指明用户是否可以编辑 JTextField。
- setHorizontalAlignment 指明 JTextField 内文本的对齐方式。
- setText 指明 JTextField 中所显示的文本。

教程 4：完整的库存清单应用程序（引入程序设计的概念）

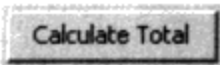
Integer Integer 类中含有一些处理整数值的方法。

● 方法

- parseInt 返回一个同字符串参数等价的整数。

JButton 使用户通过点击应用程序图形用户界面中的按钮，来产生一种事件。

● 运行



● 事件

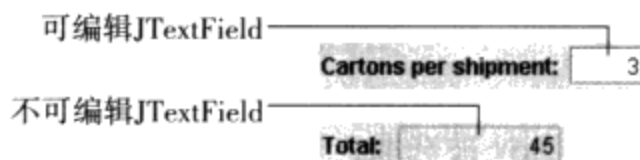
actionPerformed 用户点击 JButton 时执行。

● 方法

- setBounds 设置 JButton 的位置与大小。
- setText 设置 JButton 的文本属性。

TextField 该组件允许用户输入信息，同时也用于向用户显示结果。

● 运行



● 方法

getText 返回 JTextField 中所显示的文本。

setBounds 设置 JTextField 的位置与大小。

setEditable 指明用户是否能够编辑 JTextField。若为 true（默认值），则意味 JTextField 是一个可编辑的输入 JTextField，若为 false，则意味 JTextField 是一个不可编辑的输出 JTextField。

setHorizontalAlignment 指定 JTextField 中文本的对齐方式（可以选取的值：JTextField.LEFT，JTextField.CENTER，JTextField.RIGHT）。

setText 指定 JTextField 中将显示的文本。

String String 类用于存储和管理文本数据。

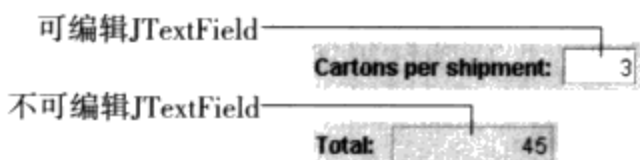
● 方法

valueOf 返回与参数值等价的字符串。

教程 5：改进的库存清单应用程序（引入变量、内存、算术运算及键盘事件概念）

TextField 该组件允许用户输入信息，同时也可用来向用户显示结果。

● 运行



● 事件

keyPressed 任何作用在 JTextField 中的按键会产生此事件。

● 方法

getText 返回 JTextField 中显示的文本。

setBounds 设置 JTextField 的位置与大小。

setEditable 指明用户是否能够编辑 JTextField。若为 true（默认值），则意味 JTextField 是一个可编辑的输入 JTextField，若为 false，则意味 JTextField 是一个不可编辑的输出 JTextField。

setHorizontalAlignment 指定 JTextField 中文本的对齐方式。

setText 指定 JTextField 中将显示的文本。

教程 6：工资额计算器应用程序（引入算法、伪代码及程序控制的概念）

DecimalFormat DecimalFormat 类用于格式化浮点数（即带有小数点的数）。

● 方法

format 能够将一个 double 值转换为指定的格式。

教程 7：牙科付款应用程序（介绍 JCheckBox，消息对话框和逻辑运算符）

JCheckBox 该组件允许用户选择某个选项。如果应用程序中含有多个 JCheckBox，则可一次选择任意数目的 JCheckBox（0 个或全部）。

[G e n e r a l I n f o r m a t i o n]

书名= J A V A大学简明教程：实例程序设计

作者= (美) H . M . D e i t e l 等著；张琛恩等译

页数= 8 3 5

出版社= 北京市：电子工业出版社

出版日期= 2 0 0 5 . 0 2

S S 号= 1 1 6 8 8 6 2 5

D X 号= 0 0 0 0 0 5 3 3 2 7 0 3

URL = h t t p : / / b o o k . s z d n e t . o r g . c n / b o o k D e t a i l . j s
p ? d x N u m b e r = 0 0 0 0 0 5 3 3 2 7 0 3 & d = A 2 F 6 0 3 A 7 6 7 1 0 4 F E 1 4
A F 3 5 C D 1 D 5 D 7 E F D B

封面		
前言		
教程 1	M o v i n g S h a p e s 应用程序	介绍计算机，I n t e r n e t 及 J a v a
程序设计基础		
1 . 1	什么是计算机	
1 . 2	计算机的组织结构	
1 . 3	机器语言、汇编语言和高级语言	
1 . 4	J a v a 概述	
1 . 5	其他高级语言	
1 . 6	结构化程序设计	
1 . 7	诠释软件的发展方向：对象技术	
1 . 8	I n t e r n e t 与万维网	
1 . 9	J a v a 运行环境	
1 . 1 0	新手上路：M o v i n g S h a p e s 应用程序	
1 . 1 1	I n t e r n e t 及 W e b 资源	
1 . 1 2	小结	
教程 2	W e l c o m e 应用程序	引入图形用户界面的程序设计
2 . 1	探试 W e l c o m e 应用程序	
2 . 2	编译并运行模板 W e l c o m e 应用程序	
2 . 3	创建 W e l c o m e 应用程序	
2 . 4	语法错误	
2 . 5	小结	
教程 3	库存清单应用程序	介绍 J T e x t F i e l d 和 J B u t t o n 组件
3 . 1	探试库存清单应用程序	
3 . 2	在库存清单应用程序中自定义 J L a b e l	
3 . 3	自定义库存清单应用程序的 J T e x t F i e l d 和 J B u t t o n	
3 . 4	小结	
教程 4	完整的库存清单应用程序	引入程序设计的概念
4 . 1	探试库存清单应用程序	
4 . 2	介绍 J a v a 的代码规则	
4 . 3	在事件处理程序中放置代码	
4 . 4	执行计算并显示结果	
4 . 5	小结	
教程 5	改进的库存清单应用程序	引入变量、内存、算术运算及键盘事件的概念
5 . 1	探试改进后的库存清单应用程序	
5 . 2	变量	
5 . 3	针对 J T e x t F i e l d 处理 k e y P r e s s e d 事件	
5 . 4	内存的概念	
5 . 5	算术运算	
5 . 6	调试程序：断点设置与 r u n , s t o p , c o n t 和 p r i n t 命令	
5 . 7	I n t e r n e t 与 W e b 资源	
5 . 8	小结	
教程 6	工资额计算器应用程序	引入算法、伪代码及程序控制的概念
6 . 1	探试工资额计算器应用程序	
6 . 2	算法	
6 . 3	伪代码	
6 . 4	控制语句	
6 . 5	i f 选择语句	
6 . 6	i f ... e l s e 选择语句	
6 . 7	创建工资额计算器应用程序	
6 . 8	赋值运算符	
6 . 9	格式化文本	
6 . 1 0	使用调试程序：p r i n t 命令和 s e t 命令	
6 . 1 1	小结	
教程 7	牙科付款应用程序	介绍 J C h e c k B o x , 消息对话框和逻辑运算符

7 . 1	探试牙科付款应用程序	
7 . 2	创建牙科付款应用程序	
7 . 3	学习使用 J C h e c k B o x	
7 . 4	使用对话框显示消息	
7 . 5	逻辑运算符	
7 . 6	小结	
教程 8	购车还贷计算器应用程序	介绍 w h i l e 循环语句及 J T e x t A r e a 组件
8 . 1	探试购车还贷计算器应用程序	
8 . 2	w h i l e 循环语句	
8 . 3	自增运算符和自减运算符	
8 . 4	创建购车还贷计算器应用程序	
8 . 5	小结	
教程 9	班级平均分应用程序	介绍 d o ... w h i l e 循环语句
9 . 1	探试班级平均分应用程序	
9 . 2	d o ... w h i l e 循环语句	
9 . 3	创建班级平均分应用程序	
9 . 4	小结	
教程 10	利息计算器应用程序	介绍 f o r 循环语句
10 . 1	探试利息计算器应用程序	
10 . 2	计数器控制循环的要素	
10 . 3	引入 f o r 循环语句	
10 . 4	f o r 循环举例	
10 . 5	创建利息计算器应用程序	
10 . 6	小结	
教程 11	门禁系统应用程序	介绍 s w i t c h 多向选择语句, D a t e 及 D a t e F o r m a t 类
11 . 1	探试门禁系统应用程序	
11 . 2	介绍 s w i t c h 多向选择语句	
11 . 3	创建门禁系统应用程序	
11 . 4	小结	
教程 12	改进的工资额计算器应用程序	引入方法的概念
12 . 1	探试改进的工资额计算器应用程序	
12 . 2	类与方法	
12 . 3	方法的定义	
12 . 4	最大值应用程序	
12 . 5	在工资额计算器应用程序中使用方法	
12 . 6	使用调试程序: 利用 s t e p , s t e p u p 和 n e x t 命令控制执行	
12 . 7	小结	
教程 13	改进的利息计算器应用程序	引入事件处理的概念
13 . 1	探视改进的利息计算器应用程序	
13 . 2	事件处理程序	
13 . 3	事件处理程序的注册	
13 . 4	处理 C h a n g e E v e n t 事件	
13 . 5	小结	
教程 14	筹款募集应用程序	介绍作用域及基本类型间的转换
14 . 1	探试筹款募集应用程序	
14 . 2	创建筹款募集应用程序	
14 . 3	类型转换	
14 . 4	小结	
教程 15	掷骰子游戏应用程序	介绍随机数的生成和 J P a n e l 组件
15 . 1	探试掷骰子游戏应用程序	
15 . 2	随机数的生成	
15 . 3	在掷骰子游戏应用程序中使用常量	
15 . 4	在掷骰子游戏应用程序中使用随机数	
15 . 5	小结	

教程 1 6	国旗知识测评应用程序	介绍一维数组及 J C o m b o B o x 组件
1 6 . 1	探试国旗知识测评应用程序	
1 6 . 2	数组	
1 6 . 3	声明并创建数组	
1 6 . 4	创建国旗知识测评应用程序	
1 6 . 5	数组的排序	
1 6 . 6	小结	
教程 1 7	成绩评定应用程序	介绍二维数组及 J R a d i o B u t t o n 组件
1 7 . 1	探试成绩评定应用程序	
1 7 . 2	二维数组	
1 7 . 3	学习使用 J R a d i o B u t t o n	
1 7 . 4	在成绩评定应用程序中添加代码	
1 7 . 5	小结	
教程 1 8	微波炉模拟应用程序	
	创建属于自己的类及其对象	
1 8 . 1	探试微波炉模拟应用程序	
1 8 . 2	设计微波炉模拟应用程序	
1 8 . 3	对象的初始化：构造方法	
1 8 . 4	g e t 方法和 s e t 方法	
1 8 . 5	完成微波炉模拟应用程序	
1 8 . 6	控制成员的访问	
1 8 . 7	m a i n 方法	
1 8 . 8	使用调试程序：w a t c h 命令	
1 8 . 9	小结	
教程 1 9	货运中心应用程序	介绍集合，A r r a y L i s t 及迭代器
1 9 . 1	探试货运中心应用程序	
1 9 . 2	P a r c e l 类	
1 9 . 3	J L i s t 组件	
1 9 . 4	使用助记符	
1 9 . 5	集合	
1 9 . 6	创建货运中心应用程序	
1 9 . 7	使用迭代器	
1 9 . 8	小结	
教程 2 0	屏保应用程序	引入继承及图形绘制概念
2 0 . 1	探试屏保应用程序	
2 0 . 2	继承简介	
2 0 . 3	图形绘制简介	
2 0 . 4	创建屏保应用程序	
2 0 . 5	利用继承创建 M y R e c t a n g l e 类	
2 0 . 6	J a v a 中的图形绘制	
2 0 . 7	完成屏保应用程序	
2 0 . 8	小结	
教程 2 1	“猫 - 鼠”小画家应用程序	介绍接口、鼠标输入及事件处理机制
2 1 . 1	探试小画家应用程序	
2 1 . 2	创建小画家应用程序	
2 1 . 3	接口	
2 1 . 4	m o u s e P r e s s e d 事件处理程序	
2 1 . 5	m o u s e R e l e a s e d 事件处理程序	
2 1 . 6	m o u s e D r a g g e d 事件处理程序	
2 1 . 7	小结	
教程 2 2	打字训练器应用程序	介绍键盘事件及 J M e n u 组件
2 2 . 1	探试打字训练器应用程序	
2 2 . 2	键盘事件	
2 2 . 3	J M e n u	
2 2 . 4	J C o l o r C h o o s e r	

	2 2 . 5	小结	
教程 2 3	屏幕抓取应用程序	介绍字符串处理技术	
	2 3 . 1	探试屏幕抓取应用程序	
	2 3 . 2	字符串基础	
	2 3 . 3	创建屏幕抓取应用程序	
	2 3 . 4	在字符串中定位子字符串	
	2 3 . 5	从字符串中提取子字符串	
	2 3 . 6	S t r i n g 类中的其他方法	
	2 3 . 7	小结	
教程 2 4	改进的购车还贷计算器应用程序	介绍异常处理技术	
	2 4 . 1	探试改进的购车还贷计算器应用程序	
	2 4 . 2	介绍异常处理技术	
	2 4 . 3	J a v a 中的异常处理	
	2 4 . 4	J a v a 中异常的结构	
	2 4 . 5	创建改进的购车还贷计算器应用程序	
	2 4 . 6	小结	
教程 2 5	票务信息查询应用程序	介绍按顺序存取的文件	
	2 5 . 1	探试活动录入应用程序和票务信息查询应用程序	
	2 5 . 2	数据分级	
	2 5 . 3	文件和流	
	2 5 . 4	创建活动录入应用程序：向文件中写入信息	
	2 5 . 5	创建票务信息查询应用程序	
	2 5 . 6	学习使用 f i n a l l y 语句块	
	2 5 . 7	小结	
教程 2 6	A T M 应用程序	介绍数据库程序设计及命令行参数的使用	
	2 6 . 1	I B M C l o u d s c a p e 数据库	
	2 6 . 2	探试 A T M 应用程序	
	2 6 . 3	设计 A T M 应用程序	
	2 6 . 4	关系型数据库纵览：A T M 数据库	
	2 6 . 5	S Q L	
	2 6 . 6	命令行参数的使用	
	2 6 . 7	创建数据库连接	
	2 6 . 8	编程实现 A T M 应用程序	
	2 6 . 9	小结	
教程 2 7	绘图应用程序	引入多态概念，进一步讨论图形绘制	
	2 7 . 1	探试绘图应用程序	
	2 7 . 2	多态	
	2 7 . 3	更多 G r a p h i c s 类的方法	
	2 7 . 4	添加 M y S h a p e 类的继承结构	
	2 7 . 5	小结	
教程 2 8	电话号码簿应用程序	J a v a 语音 A P I 简介	
	2 8 . 1	J a v a 语音 A P I	
	2 8 . 2	下载并安装 F r e e T T S	
	2 8 . 3	探试电话号码簿应用程序	
	2 8 . 4	创建电话号码簿应用程序	
	2 8 . 5	小结	
教程 2 9	W e b 书店应用程序	W e b 应用程序的开发及 A p a c h e T o m c a t W e b 服务器简介	
	2 9 . 1	多层架构	
	2 9 . 2	W e b 服务器	
	2 9 . 3	A p a c h e T o m c a t W e b 服务器	
	2 9 . 4	探试 W e b 书店应用程序	
	2 9 . 5	小结	
教程 3 0	W e b 书店应用程序：客户层	H T M L 简介	
	3 0 . 1	分析 W e b 书店应用程序	

30.2	创建JavaServer Pages
30.3	创建books.jsp页面
30.4	创建bookInformation.jsp页面
30.5	小结
教程31	Web书店应用程：信息层 考察数据库并创建数据库组件
31.1	回顾Web书店应用程序
31.2	信息层：数据库
31.3	在JSP页面中使用Cloudscape数据库
31.4	小结
教程32	Web书店应用程序：中间层 介绍form属性：method/action
和在JSP中插入查询结果集	
32.1	回顾Web书店应用程序
32.2	在books.jsp页面中添加功能
32.3	在bookInformation.jsp页面中添加功能
32.4	Internet与Web资源
32.5	小结
附录A	运算符优先级表
附录B	ASCII字符集
附录C	GUI设计导航
附录D	Java类库索引
附录E	关键字列表
附录F	基本类型
词汇表	
索引	